VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS

LABORATORNÍ ÚLOHY OSVĚTLUJÍCÍ PRINCIP SÍŤOVÝCH TECHNOLOGIÍ A PROTOKOLŮ

LABORATORY EXERCISES EXPLAINING THE PRINCIPLES OF NETWORK TECHNOLOGIES AND PROTOCOLS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. Tomáš Hricko**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Lukáš Langhammer, Ph.D.**
SUPERVISOR

BRNO 2020

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní obor **Telekomunikační a informační technika**

Ústav telekomunikací

Student: Bc. Tomáš Hricko ***ID:*** 186088
Ročník: 2 ***Akademický rok:*** 2019/20

NÁZEV TÉMATU:

Laboratorní úlohy osvětlující princip síťových technologií a protokolů

POKYNY PRO VYPRACOVÁNÍ:

Náplň diplomové práce spočívá v návrhu dvou nových laboratorních úloh ve vhodném simulačním prostředí. Nastudujte problematiku komunikačních protokolů a při výběru řešené problematiky se zaměřte na témata jako jsou transportní protokoly TCP, UDP, SCTP, porovnání směrovacích protokolů, demonstrace funkce QoS, srovnání technologií ATM a Frame Relay, objasnění funkce síťových prvků, atd. Laboratorní úlohy budou zahrnovat kompletní návody vhodné pro studenty, výchozí scénář, doplňující úkoly a kontrolní otázky. Časová náročnost každé úlohy musí být přibližně dvě hodiny.

DOPORUČENÁ LITERATURA:

[1] FOROUZAN, B. A. TCP/IP Protocol Suite. Fourt hedition, Boston: McGraw-Hill Higher Education, 2010, 979 stran. ISBN 978-0-07-337604-2.

[2] JEŘÁBEK, J. Komunikační technologie. Skriptum FEKT Vysoké učení technické v Brně, 2016. s. 1-172.

Termín zadání: 3.2.2020 ***Termín odevzdání:*** 1.6.2020

Vedoucí práce: Ing. Lukáš Langhammer, Ph.D.

prof. Ing. Jiří Mišurec, CSc.
předseda oborové rady

Abstrakt

Táto diplomová práca sa zaoberá vytvorením dvoch laboratórnych úloh pre študentov. Prvá úloha je zameraná na porovnanie transportných protokolov TCP a UDP. Druhá úloha je zameraná na porovnanie smerovacích protokolov OSPF a RIP. V prvej časti práce je popísané použité simulačné prostredie. V ďalšej časti je popísaný použitý operačný systém v smerovačoch, ďalej je popísaný postup pripravenia virtuálnych strojov. V ďalšej časti sú teoreticky opísané protokoly TCP, UDP, OSPF a RIP. Posledné dve kapitoly sú samotné laboratórne úlohy.

Kľúčové slová

GNS3, Wireshark, TCP, UDP, HTTP, TFTP, Linux, CentOS, Ubuntu, Python, VyOS, OSPF, RIP

Abstract

The aim of this diploma thesis is to create two exercises for students. The first exercise is concerned with comparison of transport protocols TCP and UDP. The second exercise is concerned with comparison of routing protocols OSPF and RIP. The first part describes used simulation environment. The next part describes the operating system used in routers. The following part describes the procedure of installing virtual machines. The next part consists of theory of protocols TCP, UDP, OSPF and RIP. The last two chapters are the laboratory exercises.

Keywords

GNS3, Wireshark, TCP, UDP, HTTP, TFTP, Linux, CentOS, Ubuntu, Python, VyOS, OSPF, RIP

Bibliografická citace:

HRICKO, Tomáš. *Laboratorní úlohy osvětlující princip síťových technologií a protokolů.* Brno, 2020. Dostupné také z: https://www.vutbr.cz/studenti/zav-prace/detail/126016. Diplomová práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedoucí práce Ing. Lukáš Langhammer, Ph.D..

Vyhlásenie

Vyhlasujem, že svoju diplomovú prácu na tému „Laboratorní úlohy osvětlující princip síťových technologií“ som vypracoval samostatne pod vedením vedúceho diplomovej práce, s využitím odbornej literatúry a ďalších informačných zdrojov, ktoré sú všetky citované v práci a uvedené v zozname literatúry na konci práce.

Ako autor uvedenej diplomovej práce ďalej vyhlasujem, že v súvislosti s vytvorením tejto diplomovej práce som neporušil autorské práva tretích osôb, najmä som nezasiahol nedovoleným spôsobom do cudzích autorských práv osobnostných a/alebo majetkových a som si plne vedomý následkov porušenia ustanovenia § 11 a nasledujúcich autorského zákona Českej republiky č. 121/2000 Sb., o práve autorskom, o právach súvisiacich s právom autorským a o zmene niektorých zákonov (autorský zákon), v znení neskorších predpisov, vrátane možných trestnoprávnych dôsledkov vyplývajúcich z ustanovenia časti druhej, hlavy VI. diel 4 Trestného zákonníka Českej republiky č. 40/2009 Sb.

V Brne dňa: ..... ……

podpis autora

Poďakovanie

Chcel by som poďakovať vedúcemu práce Ing. Lukášovi Langammerovi, Ph.D. za konzultácie a odborné vedenie a pomoc, keď som potreboval.

V Brne dňa: ..... ……

podpis autora

Obsah

Zoznam symbolov a skratiek

Skratky:

ACK	Acknowledge
BDR	Backup Designated Router
BGP	Broder Gateway Protocol
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name System
DR	Designated Router
EGP	Exterior Gateway Protocol
EIGRP	Enhanced Interior Gateway Routing Protocol
FIN	Terminate the connection
FTP	File Transfer Protocol
GNS3	Graphical Network Simulator 3
HTTP	Hypertext Transfer Procotol
HTTPS	Hypertext Transfer Protocol Secure
IP	Internet Protocol
LSA	Link State Acknowledge
LSR	Link State Request
LSU	Link State Update
OSPF	Open Shortest Path First
PSH	Push function
RIP	Routing Information Protocol
RST	Reset the connection
SMTP	Simple Mail Transfer Protocol
SYN	Synchronize the connection
TCP	Transmission Control Protocol
TFTP	Trivial File Network Protocol
UDP	User Datagram Protocol
URG	Urgent
VM	Virtual Machine
VoIP	Voice over Internet Protocol

VPN	Virtual Private Network

Symboly:

b	bit
kb	kilobit
Mb	megabit
B	bajt
kB	kilobajt
MB	megabajt

Zoznam obrázkov

Zoznam tabuliek

1. ÚVOD

Protokoly TCP (Transmission Control Protocol) a UDP (User Datagram Protocol) sú najpoužívanejšie protokoly transportnej vrstvy a bez nich by by komunikácia po internete nemohla fungovať. Využívajú sa pri rade aplikačných protokolov ako HTTP (Hypertext Transfer Protocol), FTP (File Transfer Protocol), DNS (Domain Name System), DHCP (Dynamic Host Configuration Protocol), TFTP (Trivial File Transfer Protocol) a podobne. Väčšina ľudí si ani neuvedomuje, že denno denne tieto protokoly využívajú pri komunikácii po internete.

Táto práca sa zaoberá vytvorením laboratórnej úlohy pre porovnanie protokolov TCP a UDP v rôznych scenároch.

V prvej časti úlohy je popísané simulačné prostredie a analyzačný program. V ďalšej časti práce sú popísané samotné transportné protokoly TCP a UDP a v ďalšej časti je samotné zadanie laboratórnej úlohy.

Ďalšia časť práce sa zaoberá vytvorením laboratórnej úlohy na porovnanie smerovacích protokolov. Konkrétne sa porovnávajú smerovacie protokoly OSPF (Open Shortest Path First) a RIP (Routing Information Protocol) vo verzii 2. Úloha je z väčšej časti zameraná na protokol OSPF, ktorý je podstatne zložitejší ako protokol RIP.

2. SIMULAČNÉ A ANALYZAČNÉ PROGRAMY

2.1 Simulačný program GNS3

GNS3 (Graphical Network Simulator 3) je simulačný nástroj využívaný na emuláciu, konfiguráciu, simuláciu a na odstraňovanie chýb vo virtuálnych a reálnych siatiach. GNS3 umožňuje simulovať malú topológiu obsahujúcu pár zariadení na notebooku [1]. GNS3 je voľne dostupný a je ho možné zadarmo stiahnuť z http://gns3.com. Je to aktívne vyvíjaný program a má dostupnú podporu a má stále rastúcu komunitu s viac ako 800.000 členmi [1].

2.1.1 Architektúra programu

GNS3 je zložený z dvoch softvérových komponentov [1]:

- GNS3-all-in-one software (všetko v jednom)
- GNS3 virtual machine (virtuálny stroj)

GNS3-all-in-one je klientska časť GNS3 a grafické prostredie. Program sa jednoducho nainštaluje na počítač a hneď sa dajú vytvárať vlastné sieťové topológie. Po vytvorení topológii, zariadenia, ktoré sú v topológii musia byť spustené na serverovom procese. Je viac možnosti ako spustiť serverovú časť programu [1]:

- Lokálny GNS3 server
- Lokálna GNS3 VM
- Vzdialená GNS3 VM

Lokálny GNS3 server je spustený priamo na hosťovskom počítači. Je možné ho tiež spustiť v GNS3 virtuálnom stroji, ktorý je možné spustiť na virtualizačnom softvéry ako je napríklad VirtualBox alebo VMware Workstation, poprípade na vzdialenom servery.

2.1.2 Porovnanie sieťových simulátorov

V dnešnej dobe máme k dispozícii viacero simulačných nástrojov, ktoré slúžia na simulovanie sieti. Dnes je to pre záujemcov, ktorí chcú simulovať siete je oveľa jednoduchšie ako to bolo v minulosti, vtedy mali len jednu možnosť a to buď si zariadenia kúpiť alebo si ich požičať [1].

Máme k dispozícii tieto simulačné programy:

- ns-3
- GNS3
- Cisco Packet Tracer
- Cisco VIRL
- Fyzické zariadenia
- Iné

ns-3:
ns-3 je simulátor určený najmä pre výskum a na vyučovacie účely. ns-3 je program s otvoreným kódom. ns-3 poskytuje modely ako sa paketové dáta v sieti pohybujú a ako fungujú. Poskytuje simulačný nástroj pre užívateľov pre simulovanie experimentov. Jeden z dôvodov prečo používať ns-3 je pre výskum, ktorý je veľmi zložitý alebo nemožný s reálnymi systémami [2].

GNS3:
Ako už bolo zmienené, tak GNS3 je voľné dostupný softvér s otvoreným zdrojovým kódom, ktorý je dostupný na GitHub-e [1].
Výhody GNS3 [1]:

- Je zadarmo a má otvorený zdrojový kód
- Neobmedzené množstvo simulovaných zariadení, obmedzenie je len v hardvéry, na ktorom simulujeme
- Podporuje všetky dostupné IOS obrazy, na emulovanie Cisco hardware sa používa Dynamips [3].
- Podporuje simuláciu zariadení od rôznych dodávateľov
- Môže byť použitý s virtuálizáciou hardvéru alebo bez nej
- Kompatibilita s Linuxom bez inštalovania ďalších programov
- Softvér od viacerých dodávateľov zadarmo k stiahnutiu
- Veľká komunita s viac ako 800 000 členmi

Nevýhody GNS3 [1]:

- Obrazy pre Cisco zariadenia musia byť dodané užívateľom.
- Môže byť obmedzený nastavením firewallu a ostatných zabezpečovacích nastavení

Cisco Packet Tracer:
Je to oficiálny produkt Cisco, ktorý je dostupný pre študentov Cisco Academy. Je to voľne dostupný [1].

- Výhody [1]:
- Jednoduchý na inštaláciu
- Podporuje Cisco smerovače, prepínače a simuláciu počítača
- Zadarmo, ale je potrebná registrácia na Cisco web NetAcad.

Nevýhody [1]:

- Nemá otvorený zdrojový kód
- Cisco zariadenia sú iba simulované, nebežia na ňom reálne IOS obrazy
- Nepodporuje zariadenia ostatných výrobcov
- Nedajú sa do neho zakomponovať reálne zariadenia

Cisco VIRL:
Simulačný nástroj vyvinutý spoločnosťou Cisco, ktorý na rozdiel od Cisco Packet Tracer neslúži len na učenie ale je na ňom možné simulovať reálne siete [1].
Výhody [1]:

- Podporuje Cisco smerovače, prepínače, firewally a simulácia počítača
- Vhodný na štúdium CCNA, CCNP a CCIE
- Podporuje veľké množstvo protokolov ako sú napríklad RPVST+ (Rapid per-VLAN spanning tree), Etherchannel, Port Security a ďalšie

Nevýhody [1]:

- Nie je zadarmo

- Limitované množstvo zariadení v topológii simulovanej siete
- Môže byť komplikovaný na nastavenie a konfiguráciu
- Je náročný na pamäť a procesor počítača
- Vyžaduje virtualizačný sofvér (VMware Workstation Player/Pro, Fusion, ESXI)
- Nepodporuje VirtualBox
- Nepodporuje zariadenia od iných výrobcov – podporuje iba Cisco zariadenia

2.1.3 Inštalácia programu GNS3

V našom prípade budeme program GNS3 používať na operačnom systéme Windows a na virtualizáciu prvkov budeme používať lokálny GNS3 VM virtuálny stroj. Inštalačný balíček je voľne dostupný na stiahnutie z webovej stránky https://www.gns3.com/.

Podporované verzie operačného systému Windows [1]:

- Windows 7 SP1 (64 bit)
- Windows 8 (64 bit)
- Windows 10 (64 bit)
- Windows Server 2012 (64 bit)
- Windows Server 2016 (64 bit)

Minimálne požiadavky pre program GNS3 sú v tabuľke 2-1.

Tab. 2-1: Minimálne požiadavky programu GNS3 [1]

Vec	**Požiadavka**
Operačný systém	Windows 7 (64 bit) alebo vyšší
Procesor	dve alebo viac logických jadier
Virtualizácia	Podporovaná virtualizácia
Pamäť RAM	4 GB
Pamäť	1 GB (Inštalácia na Windows je menšia ako 200 MB)

Stiahneme GNS3 all-in-one inštalačný balíček a spustíme ho:

- Potvrdíme licenčné podmienky
- Zvolíme názov skratky v Štart menu

Zvolíme potrebný voliteľný softvér:

- WinPCAP – na prepojenie GNS3 s internetom. Obsahuje prvky NAT a CLOUD
- Wireshark – program potrebný na odchytávanie komunikácie
- QEMU – slúži na virtualizáciu virtuálnych počítačov
- VPCS – jednoduchý virtuálny počítač
- Cpulimit (odporúčané) – zabraňuje QEMU aby využil procesor na 100%
- GNS3 – samotný GNS3 program
- TightVNC Viewer – VNC klient na pripojenie ku grafickému prostrediu virtuálnych strojov
- Solar-Putty – aplikácia na pripojenie k prvkom

Ďalej zvolíme miesto inštalácie a nainštalujeme program.

2.1.4 GNS3 VM – virtuálny stroj

Na virtualizáciu prvkov v topológiách budeme využívať GNS3 VM, ktorá je voľne dostupná na https://www.gns3.com/. Budeme využívať verziu pre program VMware Workstation. V tejto diplomovej práci sa bude využívať upravená GNS3 VM už s nainštalovanými prvkami (kvôli veľkosti súboru nie je v prílohe. Je dostupná u vedúceho práce).

Postup na pridanie GNS3 VM do vmware Workstation:

- Spustíme program VMware Workstation
- Vložiť virtuálny stroj do VMware Workstation. Player -> File -> Open a nájdeme v počítači stiahnutý virtuálny stroj

Virtuálny stroj GNS3 VM by mal byť dostupný v programe VMware Workstation.

Odporúčané nastavenia GNS3 VM:

- 6 GB RAM
- 4 virtuálne jadra procesoru

Vloženie GNS3 VM do programu GNS3:

- Spustíme program a v hornej lište klikneme na Help -> Setup Wizard (ak je program spustený prvý krát Setup Wizard sa spustí automaticky)
- Zvolíme „Run appliances in a virtual machine“ a klikneme na Next
- Nastavenia „Local server configuration“ sa nebudú meniť a stlačíme dva krát za sebou Next.
- Na stránke GNS3 VM zvolíme virtualizačný software VMware a zo zoznamu vyberieme stiahnutý virtuálny stroj GNS3 VM a vyberieme počet virtuálnych jadier a veľkosť pamäte RAM.

Ak program GNS3 vypisuje chybu, že nie je možné sa pripojiť k virtuálnemu stroju, tak je potrebné ešte naviac k programu VMware Workstation nainštalovať aj program VMware VIX. Predtým je potrebné zistiť verziu programu VMware Workstation a k tomu stiahnuť odpovedajúcu verziu VMware VIX. V tomto prípade bola využitá verzia programu VMware Workstation 15.5.1 a verzia VMware VIX 1.17.0. Postup na nastavenie VMware VIX:

- Prejdeme do priečinku kde je nainštalovaný program VMware VIX.
- Vytvoríme zálohu súboru vixwrapper-config.txt
- Otvoríme súbor vixwrapper-config.txt a nájdeme nasledujúce tri riadky:

```
# latest un-versioned
ws        19  vmdb  e.x.p Workstation-14.0.0
player    19  vmdb  e.x.p Workstation-14.0.0
```

- Písmena e.x.p nahradíme verziou programu VMware Workstation, v tomto prípade je to 15.5.1 a uložíme zmeny v súbore:

```
# latest un-versioned
ws        19  vmdb  15.5.1 Workstation-14.0.0
player    19  vmdb  15.5.1 Workstation-14.0.0
```

- Zopakujeme prechádzajúce kroky pre vloženie GNS3 VM do GNS3.

3. OPERAČNÉ SYSTÉMY SIEŤOVÝCH PRVKOV

Na to aby sieťové prvky ako sú smerovače prepínače, firewally a podobne mohli fungovať potrebujú mať operačný systém. Operačný systém týchto prvkov sa väčšinou ovláda pomocou príkazového riadku a takisto sa prvky konfigurujú pomocou príkazového riadku. Vo svete je dostupných viacero operačných systémov pre sieťové prvky, od tých, ktoré sú dostupné zdarma a tie, ktoré sú dostupné len od konkrétnych výrobcov.

Najznámejšie operačný systémy:

- Cisco IOS – je to operačný systém firmy Cisco, ktorý sa využíva v zariadeniach firmy Cisco. Systém je proprietárny, takže ho nie je možné využívať na iných zariadeniach ako sú zariadenia Cisco. Prostredie systému je príkazový riadok.
- Junos OS – operačný systém od firmy Juniper Networks používaný v zariadeniach Juniper. Rovnako ako Cisco IOS tento operačný systém nie je voľne dostupný
- MikroTik RouterOS – operačný systém pre zariadenia s RouterBoard. RouterOS na rozdiel od prechádzajúcich zmienených operačných systémov ponúka konfiguráciu pomocou aplikácie WinBox, čo je prepracované grafické prostredie na konfiguráciu zariadení s RouterOS.
- VyOS – operačný systém s otvoreným zdrojovým kódom a voľne dostupný. Je založený na operačnom systéme Debian GNU/Linux. Prostredie operačný systému je príkazový riadok. [6]

3.1 Operačný systém VyOS

V tejto diplomovej práci sa využívajú virtuálne sieťové prvky v simulačnom prostredí GNS3, ktoré majú nainštalovaný operačný systém VyOS. Výhoda tohto operačného systému je to, že je voľne dostupný a jeho zdrojový kód je tiež voľne dostupný.

Je to sieťový operačný systém, ktorý ponúka funkcie ako smerovanie, firewall a VPN. Projekt VyOS začal v roku 2013 ako komunitná odroda Vyatta Core 6.6R1 s cieľom ponúkať otvorený a voľne dostupný sieťový operačný systém. Bola to odpoveď na rozhodnutie nepokračovania v komunitnej verzii Vyatta.

VyOS je založený na operačnom systéme Debian GNU/Linux a využíva Quagga smerovací modul. [6]

Funkcie VyOS [6]:

- Podporuje smerovacie protokoly ako sú OSPFv2, OSPFv3, BGP, RIP a ďalšie.
- Prostredie systému je príkazový riadok.
- Podporuje skriptovanie
- Stavová konfigurácia, najprv sa zadajú príkazy na zmenu konfigurácie a potom sa pomocou príkazu commit zmeny zapíšu. Pred zápisom zmien, je možné zmeny porovnať poprípade neuložiť zmeny a vrátiť sa do predošlej konfigurácie.
- Podporuje viacero VPN služieb ako sú napríklad OpenVPN, IPSec, IKEv2.
- Podporuje DHCP, TFTP.
- Podpora oboch IPv4 aj IPv6 protokolov.
- Kompatibilný s reálnym hardwarom aj s virtuálnymi strojmi.

4. PRÍPRAVA VIRTUÁLNYCH STROJOV PRE LABORATÓRNE ÚLOHY

V tejto kapitole je rozpísané aké programy sú nainštalované na virtuálnych strojoch (Virtual Machine – VM) a ako ich nainštalovať.

4.1 Operačný systém

Pre server aj pre klienta je zvolený operačný systém CentOS 8 s minimálnou inštaláciu, to znamená, že prostredie VM je len v príkazovom riadku. Operačný systém CentOS je voľné dostupný operačný systém, ktorý je dostupný z https://centos.org.

4.2 Nainštalovanie základných programov

Pre laboratórne úlohy budú potrebné nasledujúce programy:

- wget – program na sťahovanie pomocou protokolov HTTP, HTTPS, FTP
- vim, nano – programy na úpravu súborov
- traceroute – program na hľadanie cesty k cieľovému sieťovému zariadeniu
- bind-utils – obsahuje kolekciu príkazov na dotazovanie DNS serverov
- python2 – pre spustenie skriptov napísaných v jazyku Python

Pred inštaláciou je potrebné ešte aktualizovať repozitár a operačný systém, potom môžeme nainštalovať programy pomocou príkazov:

```
$ sudo dnf update -y
$ sudo dnf install wget vim nano traceroute bind-utils python2 -y
```

4.3 Príprava servera

Každý scenár laboratórnej úlohy bude mať vlastné VM. Ale VM s operačným systémom bude len jeden a ostatné budú len linkované klony, a to z dôvodu ušetrenia pamäte počítača. Na hlavný server s operačným systémom sa nainštaluje potrebný softvér a po tom som urobia linkované klony z hlavného. Potrebný softvér pre server v laboratórnych úlohách je nasledovný:

- httpd – je softvérový webový HTTP server
- tftp, tftp-server – softvérový tftp klient a server

Tieto programy nainštalujeme pomocou tohto príkazu:

```
$ sudo dnf install httpd tftp tftp-server -y
```

4.4 Príprava klinta

Rovnako ako pre servery, tak aj klient bude len jedna VM a ostatné používané v laboratórnych úlohách budú len linkované klony. Potrebný softvér pre klienta je nasledovný:

- tftp – softvér pre tftp klienta

Program sa nainštaluje pomocou príkazu:

```
$ sudo dnf install tftp -y
```

5. TRANSPORTNÁ VRSTVA

Prvá laboratórna úloha sa zaoberá porovnaním protokolov transportnej vrstvy. Na transportnej vrstve sú protokoly, ktoré slúžia na komunikáciu medzi konkrétnymi aplikáciami. Ktoré aplikácie spolu komunikujú sa rozlišujú podľa transportnej adresy, takzvaných portov. Port je 16 bitové číslo, ktoré sa bežne používa s číslami desiatkovej sústavy v rozsahu 0 až 65535. Najpoužívanejšie protokoly transportnej vrstvy sú TCP a UDP, ktoré sú popísané ďalej v tejto práci.

5.1 Protokol UDP (User Datagram Protocol)

Protokol UDP je jednoduchý protokol, ktorý zaisťuje nespoľahlivý a nespojovaný prenos dát. Je nespoľahlivý v tom, že nie je zaručené, že odosielané dáta budú prijímateľovi doručené alebo doručené v správnom poradí. Protokol má len jednoduchú kontrolu dát pomocou kontrolného súčtu, ktorý ak je nesprávny, tak sa datagram zahodí. Nedostatky, ktoré protokol má prinášajú ale viacero výhod. Medzi hlavné patrí malá réžia a malé oneskorenie. Protokol, je využívaný hlavne na prenos malých správ, u ktorých nie je kritické, ak občas dôjde k chybe [7].

Protokol poskytuje aplikáciám posielať správy medzi programami s čo najmenšími protokolovými mechanizmami [8].

5.1.1 Datagram protokolu

Protokol má záhlavie, ktoré sa skladá zo štyroch 16-bitových polí. Za záhlavím pokračujú dáta aplikácie. Datagram protokolu je zobrazený na obrázku 5-1 [7].

Bity 0-15	Bity 16-31
Zdrojový port	Cieľový port
Celková dĺžka	Kontrolný súčet
Dáta aplikácie	

Obr. 5-1: Datagram protokolu UDP

- **Zdrojový port** (source port) - číslo portu používané procesom na strane odosielateľa. Ak je odosielateľom klient, tak je číslo portu vybrané z rozsahu dynamických alebo registrovaných portov. Ak je odosielateľom server, tak je väčšinou vybraté číslo portu podľa používanej služby [5].
- **Cieľový port** (destination port) - číslo portu na strane prijímateľa. Výber čísla portu platí rovnako ako pri zdrojovom porte podľa toho či sa jedná o klienta alebo o server [5].
- **Celková dĺžka** (total length) - celková dĺžka datagramu vrátane záhlavia v bajtoch [5].
- **Kontrolný súčet** (checksum) - základná kontrolná dát na úrovní transportnej vrstvy. Kontrolný súčet UDP datagramu je počítaný aj z IP záhlavia, do ktorého bude neskôr UDP datagram zapuzdrené, nie len z UDP záhlavia a dát [5].

5.1.2 Využitie protokolu UDP

Protokol UDP sa využíva hlavne na komunikáciu otázka-odpoveď. Najlepším príkladom je napríklad protokol DNS (Domain Name System), pri ktorom klient posiela dotaz na preloženie doménového mena na IP adresu. Dotaz aj odpoveď sú krátke správy, ak dôjde k strate pri prenose, tak po krátkom čase klient zas pošle DNS dotaz.

Ďalším príkladom využitia UDP protokolu je VoIP (Voice over IP). Pri VoIP nie je až tak kritické ak sa nejaký datagram stratí, pretože jeden datagram prenáša len malú časť hlasu. Pri prenose musí byť dodržané poradie datagramov, to ale musí byť vyriešené na aplikačnej vrstve [7].

5.2 Protokol TCP (Transmission Control Protocol)

Na rozdiel od protokolu UDP protokol TCP je spoľahlivý protokol a spojovo-orientovaný protokol [9]. Zaručuje správne doručenie dát a doručenie v správnom poradí. Z tohoto dôvodu je protokol TCP zložitejší ako UDP a má viacero mechanizmov, ktoré tieto požiadavky na protokol dokážu splniť.

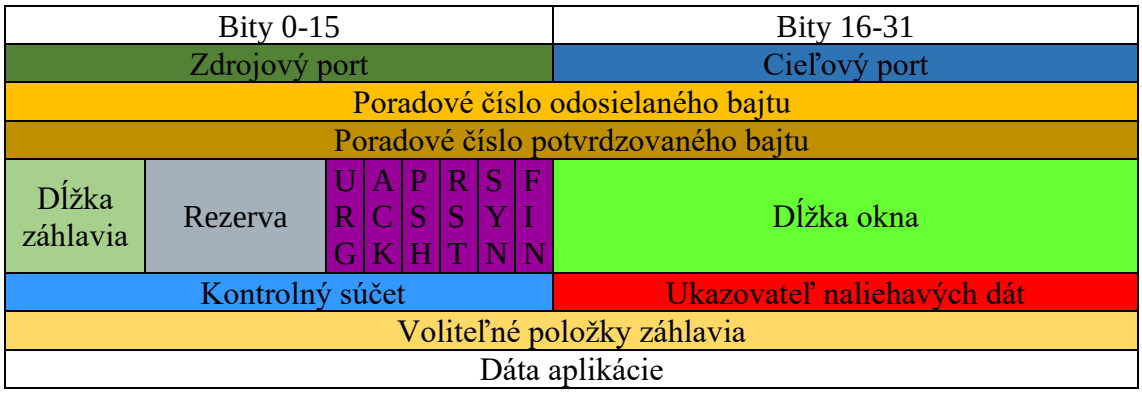
5.2.1 Vlastnosti protokolu TCP

Hlavný rozdiel medzi protokolom TCP a UDP je v tom, že pri protokole TCP sa nadväzuje spojenie a je to spoľahlivý protokol, to znamená, že je zaručené, že dáta dôjdu v správnom poradí a dôjdu všetky. Tieto vlastnosti zaručujú, spoľahlivosť protokolu [5]:

- **Číslovací systém –** v záhlaví protokolu sú číslované odosielané a prijímané bajty, nečísluje sa len jednotlivé poradie odosielaných a prijímaných segmentov.
- **Riadenie toku dát** – je založené na zmene veľkosti okna a podobných princípoch.
- **Riadenie chybových stavov** – keďže TCP je spoľahlivý protokol, tak musí mať princípy sledovania chýb a spôsoby ako na tie chyby reagovať.
- **Riadenie stavu zahltenia** – TCP dokáže reagovať na zmenu zahltenia v sieti a to zmenou množstva a rýchlosti odosielaných dát

5.2.2 Segment protokolu TCP

Ako už bolo povedané, tak protokol TCP je spoľahlivý a tak je jasné, že záhlavie protokolu bude podstatne väčšie ako pri protokole UDP. UDP záhlavie má pevnú veľkosť 8 bajtov a TCP záhlavie má premennú veľkosť záhlavia od 20 bajtov až do 60 bajtov. TCP záhlavie nasleduje po záhlaví IP (Internet Protocol) protokolu. Záhlavie protokolu je na obrázku 5-2 [7][10].



Obr. 5-2: Záhlavie TCP segmentu [7]

Význam položiek:

- **Zdrojový port** (source port) – číslo portu odosielateľa segmentu, rovnako ako pri UDP.
- **Cieľový port** (destination port) – číslo portu na strane príjemcu paketu, rovnako ako pri UDP.
- **Poradové číslo odosielaného bajtu** (sequence number - SEQ) – číslovanie odosielaných bajtov, pole obsahuje poradie prvého odosielaného bajtu.
- **Poradové číslo potvrdzovaného bajtu** (acknowledge number – ACK) – pri obojsmernej komunikácií má strana odosielateľa možnosť potvrdiť prijate dát od druhej strany. Toto pole obsahuje poradie ďalšieho odosielaného bajtu od druhej strany, takže napríklad ak príjme bajt 200, tak pole bude mať hodnotu 201.
- **Dĺžka záhlavia** (header length) – dĺžka celého záhlavia. Položka je povinná, pretože pole voliteľných položiek záhlavia má premennú dĺžku (0-40 bajtov)
- **Príznakové bity** (flags):
  - **URG** (urgent) – segment obsahuje naliehavé dáta.
  - **ACK** (acknowledge) – potvrdenie, že hodnota posledného prijatého bajtu je správna. Tým potvrdzuje prijaté dáta.
  - **PSH** (push function) – indikuje, že prijaté dáta majú byť ihneď predané aplikácii a nemá sa čakať na ďalšie segmenty.
  - **RST** (reset the connection) – pri prijatí duplikovaných segmentov, pre odmietnutie spojenia.
  - **SYN** (synchronize squence numbers) – odosielateľ začína novú sekvenciu odosielaných dát, využíva sa pri začiatku spojenia.
  - **FIN** (terminate the connection) – odosielateľ ukončuje prenos, využíva sa pre ukončenie spojenia.
  - **Dĺžka okna** (window size) – maximálna veľkosť okna pre odosielateľa. Hodnota sa môže podľa potreby meniť.

- **Kontrolný súčet** (checksum) – rovnako ako pri UDP datagrame, využíva sa na určitú kontrolu prijatých dát. Počíta sa z TCP záhlavia, dát a časti IP protokolu.
- **Ukazovateľ naliehavých dát** (urgent pointer) – ak je príznakový bit URG nastavený na 1 tak je pole vyplnené.
- **Voliteľné položky záhlavia** (options) – dĺžka poľa sa určite podľa dĺžky záhlavia, to znamená, že pole nemusí byť vôbec prítomné.

5.2.3 Využitie protokolu TCP

Protokol sa využíva na služby, ktoré potrebujú mať spoľahlivý prenos dát. Napríklad sa využíva pri protokoloch HTTP (Hypertext Transfer Protocol), FTP (File Transfer Protocol) a ďalšie. Pri týchto aplikáciách je neprístupné aby sa nejaké dáta stratili a aplikácia nemala možnosť o dáta požiadať znova [7].

6. SMEROVACIE PROTOKOLY

Smerovacie protokoly sú protokoly, ktoré slúžia na to aby smerovače vedeli kam majú poslať paket, ktorý prijali. Bez smerovacích protokolov by internet nemohol fungovať tak ako funguje.

Najpoužívanejšie smerovacie protokoly sú napríklad OSPF (Open Shortest Path First), EIGRP (Enhanced Interior Gateway Routing Protocol), BGP (Border Gateway Protocol), RIP (Routing Information Protocol) a ďalšie.

6.1 Delenie smerovacích protokolov

Protokoly sa delia na:

- Protokoly pre použitie vo vnútri autonómneho systému
- Protokoly pre použitie medzi autonómnymi systémami

Smerovacie protokoly vo vnútri autonómneho systému slúžia na to, že sa smerovacie informácie vymieňajú len vo vnútri konkrétneho autonómneho systému. Medzi tieto protokoly patria napríklad [7]:

- OSPF
- EIGRP
- RIP
- IS-IS (Intermediate System to Intermediate System)

Smerovacie protokoly, ktoré sú na použitie medzi autonómnymi systémami slúžia na výmenu smerovacích informácií medzi autonómnymi systémami. Najdôležitejší protokol, ktorý patrí do tejto skupiny je [7]:

- BGP

6.2 OSPF – Open Shortest Path First

OSPF je smerovací protokol, ktorý bol vytvorený organizáciou IETF približne v rokoch 1988 až 1991. Najnovšia verzia je definovaná v RFC2328. Protokol patrí do skupiny smerovacích protokolov IGP (Interior Gateway Routing Protocols). Používa sa vo vnútri autonómnych systémov. [11]

Protokol je predstaviteľom typu Link State, to znamená, že v pamäti smerovača sa vytvára kompletná mapa celej siete. Potom pomocou algoritmu, ktorý je označovaný ako Shortest Path First (SPF) sa prevádzajú výpočty potrebné k nájdeniu najkratšej cesty do konkrétnych sieti. Existujú ešte smerovacie protokoly typu Distance Vector, ktoré využívajú Bellmal-Fordovov algoritmus, kde si smerovače vymieňajú svoje smerovacie tabuľky, ale nemajú informácie o celej sieti. Do tejto skupiny protokolov patria napríklad protokoly RIP, IGMP [11].

6.2.1 Funkčnosť protokolu OSPF

Zjednodušená funkčnosť protokolu OSPF sa dá popísať v nasledujúcich krokoch [11]:

1. Smerovače posielajú do svojich OSPF rozhraní takzvané Hello Pakety. Ak sa dva poprípade viac smerovačov dohodne na určitých parametroch (Hello interval,

Dead interval, Area ID, Stub Flag, MTU veľkosť) tak sa stanu susedmi (neighbors).
2. Medzi niektorými susedmi sa vytvárajú užšie väzby. Tieto smerovače sa označujú ako priľahlé (adjacent).
3. Priľahlé smerovače si medzi sebou vymieňajú LSA (Link State Advertisement) informácie. Tieto informácie popisujú stav rozhrania smerovača alebo zoznam pripojených smerovačov k danej sieti.
4. Smerovače si ukladajú LSA, ktoré prijali do svojej databáze a zároveň ich preposielajú ostatným smerovačom v sieti. Takto bude na všetkých smerovačoch zhodná databáza topológie.
5. Keď je databáza kompletná tak smerovač prevedie výpočet pomocou Dijkstrovho algoritmu. Jeho výpočet bude mať následok, že sa nájdu najkratšie cesty do všetkých známych sieti a takisto odstránenie smyčiek.
6. Z vypočítaných dát sa naplní smerovacia tabuľka smerovača.
7. Ak v sieti nastane zmena, tak smerovač, kde zmena nastala rozošle priľahlým smerovačom informáciu v podobe LSA v OSPF pakete. Táto informácia sa postupne rozošle po celej sieti a všetky smerovače si upravia svoju databázu a prevedú nový SPF výpočet.

OSPF má výhodu oproti starším smerovacím protokol (napríklad RIP) v tom, že dokáže pracovať v pomerne veľkých sieťach. Je to dosiahnuté tým, že hierarchia OSPF je rozdelená do dvoch úrovní. Celá sieť je rozdelená na takzvané oblasti (area). LSA sa bežne posielajú len vo vnútri jednej oblasti a výpočet SFP sa spúšťa tiež len na úrovní oblastí. Z jednej oblasti do inej sa potom môžu posielať len sumarizácie oblastí. [11]

6.2.2 Metrika OSPF

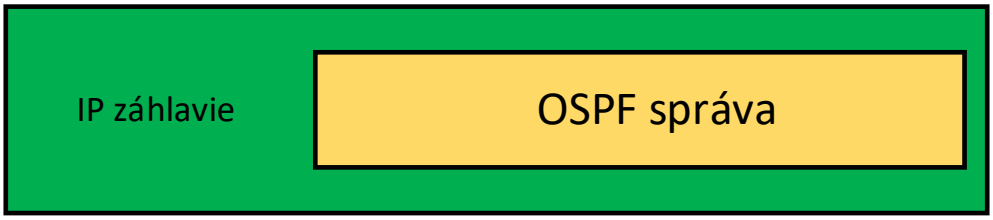
V protokole OSPF sa najkratšia cesta vyberá pomocou ceny cesty (cost). Tá môže naberať hodnoty 1 až 65535. Menšia cena cesty je preferovaná pred vyššou cenou. Cena cesty sa štandardne prideľuje k rozhraniu podľa toho akú má nastavenú šírku pásma. Cena sa dá vypočítať z nasledujúceho vzťahu [11]:

$$\text{cost} = 100000000 / \text{šírka pásma v bps} \quad (1)$$

Aby mechanizmus OSPF fungoval, tak každé rozhranie musí mať pridelenú šírku pásma. Keďže pri rozhraniach FastEthernet a pri rýchlejších by bola hodnota ceny 1, tak sa cena dá manuálne nastaviť na každé rozhranie. Výsledná cena cesty od smerovača do cieľovej siete bude súčet cien odchádzajúcich rozhraní. [11]

6.2.3 Komunikácia medzi smerovačmi

Smerovače na komunikáciu používajú IP protokol, za ktorým nasleduje OSPF hlavička, ktorá určuje typ paketu (Hello, Link State Request a podobne). Na obrázku 6-1 je ukážka zapuzdrenia protokolu OSPF v protokole IP.



Obr. 6-1: Zapuzdrenie správy OSPF v protokole IP [12]

OSPF pakety si vymieňajú len susedné smerovače. OSPF využíva na komunikáciu milticast, tým sa zabezpečí, že OSPF pakety budú prijímať len zariadenia, ktoré to podporujú.

6.2.4 Nadväzovanie susedstva medzi smerovačmi

Predtým ako si začnú smerovače vymieňať smerovacie informácie tak medzi sebou musia nadviazať susedstvo. K tomu slúžia Hello pakety, ktoré sa periodicky vysielajú na každom nakonfigurovanom rozhraní. [11]

K jednoznačnú identifikáciu medzi smerovačmi sa využíva takzvaný Router-ID. Ako Router-ID sa v základe využije najvyššia adresa nakonfigurovaná na loopback rozhraní. Ak na žiadnom loopback rozhraní nie je nakonfigurovaná žiadna adresa, tak sa využije najvyššia adresa z ktoréhokoľvek rozhrania. Bez Router-ID nemôže OSPF na smerovači fungovať. Označenie Router-ID môže mať tiež nastavenú ľubovoľnú adresu. [11]

V Hello Pakete sa posiela Router-ID, oblasť, Hello interval, Dead interval, typ oblasti. Susedstvo medzi smerovačmi sa nadviaže v prípade ak sú smerovače v rovnakej oblasti, rovnaký typ oblasti a hodnoty Hello interval a Dead interval sú rovnaké. Ak tieto podmienky nie sú splnené tak sa susedstvo nenadviaže.

V prípade, že smerovač neprijme od suseda Hello paket v Dead intervale, tak zruší susedstvo s daným susedom.

6.2.5 Výmena typologických informácií medzi smerovačmi

Po tom ako smerovače medzi sebou nadviažu susedstvo tak medzi smerovačmi vytvorené takzvané adjacency spojenie (priľahlé). Iba priľahlé smerovače si medzi sebou môžu vymieňať smerovacie informácie. [11]

Komunikácia medzi smerovačmi začína tým, že si pošlú Database Description Packet. Smerovač, ktorý má vyššie Router-ID sa stáva Master a druhý smerovač sa stáva Slave. Smerovače prijaté Database Description Pakety porovnávajú so svojou databázou a ak zistia, že im nejaká položka v databáze chýba alebo že je stará, tak si vyžiadajú aktualizáciu od druhého smerovača pomocou Link State Request (LSR) paketu. Na LSR paket smerovač odpovie Link State Update (LSU) paketom, v ktorom pošle požadované informácie. Paket LSU sa potvrdzuje Link State Acknowledgment (LSA) paketom. Ak smerovač neobdrží LSA paket po uplynutí timeoutu, tak pošle LSU paket znova. [11]

Podobne ako pri nadväzovaní spojenia komunikácia pokračuje aj keď nastane zmena v sietí (smerovač vytvorí nové susedstvo, uplynie Dead interval s nadviazaným susedom). Smerovač pošle LSU paket s touto informáciou všetkým priľahlým smerovačom. Tie to zas prepošlú svojím priľahlým smerovačom a tak sa to roznesie do celej OSPF siete. [11]

6.2.6 Typy sieti

Typy sieti OSPF protokolu môžeme rozdeliť na viacero typov a to sú [11]:

- broadcast siete
- Point to point siete
- NBMA siete
- Point to multipoint siete

V broadcast sieťach je prepojených viacero smerovačov. Ak by všetky smerovače medzi sebou vytvorili priľahlé susedstvo, tak by sa v sieti posielalo pomerne veľké množstvo informácii. Kvôli tomu sa jeden zo smerovačov poverí ako poverený smerovač (Designated router, DR) a jeho záloha záložný poverený smerovač (Backup Designated Router, BDR) v prípade výpadku povereného smerovača. Každý smerovač v broadcast sieti nadväzuje spojenie len s DR a BDR smerovačom a smerovacie informácie si vymieňa len s povereným smerovačom. V prípade výpadku povereného smerovača ho nahradzuje záložný poverený smerovač a stáva sa DR a vyberie sa nový záložný poverený smerovač.

Point to point linky slúžia na to ak sú v sieti len dva smerovače. V tomto type siete sa nevoli DR a BDR smerovač, smerovače sa automaticky stávajú priľahlými. Na komunikáciu sa využíva multicast adresa 224.0.0.5. [11]

NBMA je skratka od Non Broadcast Multicast Access. Je to v podstate sieť, ktorá je možná prepojiť viac smerovačov ale nie je v sieti možné posielať broadcast pakety a tak celá komunikácia prebieha pomocou unicastu. V sieti sa volia DR a BDR smerovače. [11]

Typ siete Point to Multipoint je sieť, v ktorej sa vytvára viacero Point to point liniek. Nedochádza k voleniu DR a BDR smerovačov. [11]

6.2.7 Typy OSPF oblastí

V protokole je viacero typov oblastí:

- Tranzitná
- Stub oblasť
- Totally stub

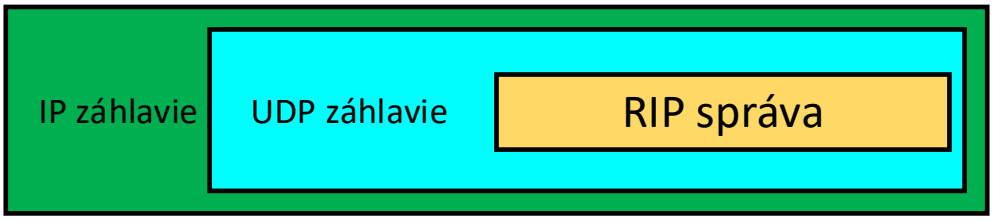
Tranzitná oblasť je oblasť cez ktorú môže prechádzať prevoz z jednej oblasti do druhej. Príklad tranzitnej oblasti je napríklad oblasť 0, tranzitnou oblasťou sa môže stať aj iná oblasť ak je v oblasti umiestnený hraničný smerovač.

Stub oblasť je typ oblasti, cez ktorú nemôže prechádzať prevoz. Ak nastavíme nejakú oblasť ako stub, tak do nej nie sú propagované externé siete, sú do nej propagované len siete zistené protokolom OSPF. Smerovanie do externých sieti je riešené takzvanou default route.

Totally stub oblasť je špeciálny typ stub oblasti. Ak zo stub oblasti existuje len jedna cesta von, taka je výhodné takúto oblasť nastaviť ako Totally stub. Všetky smerovače v tejto oblasti budú mať v smerovacej tabuľke len informácie o sieťach danej oblasti. Smerovanie k ostatným sieťam je riešené pomocou default route.

6.3 RIP – Routing Information Protocol

Protokol RIP slúži na zostavenie smerovacej tabuľky rovnako ako protokol OSPF. Protokol RIP na rozdiel od protokolu OSPF využíva na výber najkratšej cesty takzvanú metódu distance vector. Konkrétne protokol vyberá cestu s najmenším počtom skokov. Maximálny počet skokov je 15, ak je k nejakej sieti cesta dlhšia ako 15 skokov, tak protokol považuje cieľovú sieť za nedostupnú. Protokol RIP na rozdiel od protokolu OSPF využíva na transportnej vrstve protokol UDP na porte 520. Na obrázku 5-2 môžeme vidieť ako je správa RIP zapuzdrená v IP pakete na rozdiel od protokolu OSPF, ktorý sme mohli vidieť na obrázku 6-2.



Obr. 6-2: Zapuzdrenie RIP správy v IP pakete [12]

Protokol RIP ma tri verzie a to sú:

- RIPv1 – maska siete podľa triedy IP adresy
- RIPv2 – umožňuje poslať masku siete
- RIPng – RIP protokol pre protokol IPv6

Kľúčové vlastnosti protokolu RIP [12]:

- Má obmedzenú metriku na 15
- RIP správy sa posielajú v základnom nastavení pomocou broadcastu (RIPv1), multicastu (RIPv2) každých 30 sekúnd
- Ak nie je RIP správa prijatá 180 sekúnd, tak platnosť tabuľky vyprší
- Protokol nevytvára alternatívne cesty

Nevýhoda protokolu RIP je v tom, že RIP správy sa po sieti šíria veľmi pomaly a tak konvergencia smerovacích tabuliek v sietí je veľmi pomalá. [12]

7. LABORATÓRNÍ ÚLOHA – SROVNÁNÍ TRANSPORTNÍCH PROTOKOLŮ TCP A UDP

7.1 Teoretický úvod

Transportní protokoly TCP (Transmission Control Protocol) a UDP (User Datagram Protocol) jsou nejúžívanější protokoly na transportní vrstvě a jsou to důležité protokoly pro komunikaci po internetu. TCP je spojově orientovaný protokol, má datovou jednotku, které se říká segment a každý segment je potvrzovaný. TCP protokol zaručuje doručení dat ve správném pořadí, a když se paket během přenosu ztratí, pošle se znovu. Záhlaví TCP segmentu má proměnnou délku, pevná část má 20 bajtů a volitelná část má délku 0 až 40 bajtů. Pro zahájení komunikace se nejdřív naváže spojení, které se udržuje po celou dobu komunikace až po jeho ukončení. TCP protokol se využívá pro řadu služeb, které vyžadují spolehlivý protokol pro přenos dat, jako jsou například FTP (File Transfer Protocol), HTTP (Hypertext Transfer Protocol), SMTP (Simple Mail Transfer Protocol) a další. Na druhé straně protokol UDP je nespolehlivý protokol, který nezaručuje doručení dat ani správné pořadí doručení dat. Datová jednotka se v případě protokolu UDP nazývá datagram. UDP se využívá u služeb, které vyžadují malé zpoždění a kde není kritické, když se občas datagram ztratí. Využívá se například pro službu VoIP (Voice over Internet Protocol), kde jeden datagram obsahuje jen malou část slova, a proto nevadí, když se občas datagram ztratí. Při VoIP se ale vyžaduje, aby segmenty byly ve správném pořadí, avšak tuhle vlastnost protokol nesplňuje, a tak je potřeba to vyřešit na aplikační vrstvě. Protokol UDP se taky využívá při komunikaci otázka-odpověď, jako to je například při protokolu DNS (Domain Name System). Když se náhodou paket ztratí, klient odešle DNS dotaz znovu. Pro tento typ komunikace je UDP lepší než TCP také díky tomu, že na komunikaci otázka-odpověď stačí jen dva datagramy, u protokolu TCP by to bylo minimálně devět segmentů.

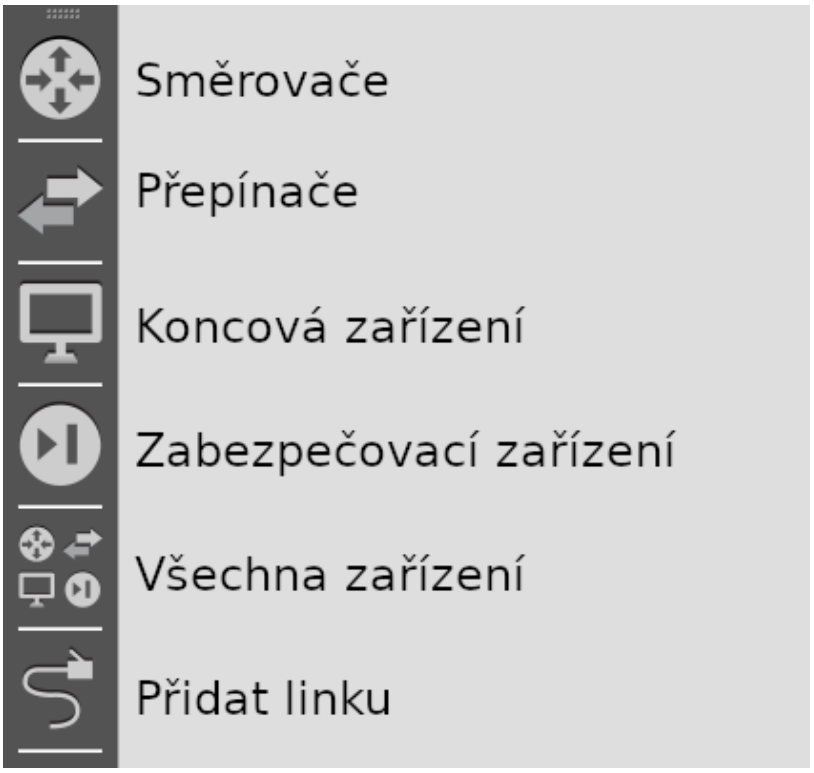
Laboratorní úloha se zabývá srovnáním TCP a UDP protokolů. V první části je všeobecné srovnání protokolů. Pro protokol TCP bude použitá služba HTTP spuštěná na Linuxovém serveru, pro protokol UDP se použije služba TFTP (Trivial File Transef Protocol). V dalších scénářích je srovnání protokolů při zahazování paketů, vypadku linky a sledování chování protokolu TCP při zvyšování provozu na lince.

7.2 Základní srovnání protokolů TCP a UDP

V první části porovnáme základní vlastnosti, jako jsou velikost záhlaví nebo kolik paketů je potřeba pro přenos malého souboru pomocí těchto protokolů.

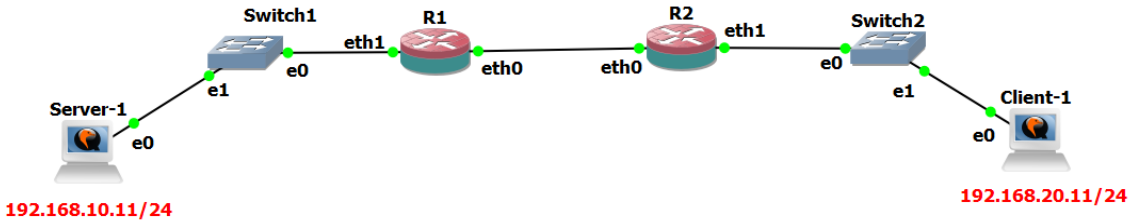
Jako první si otevřeme program GNS3 (Graphical Network Simulator 3) a vytvoříme nový projekt, který nazveme například TCPUDP. Pro simulování jsou v GNS3 přidané virtuální stroje (dále jen VM jako Virtual Machine), které mají nainstalovaný operační systém CentOS 8. V levém menu klikneme na ikonu počítače, vybereme zařízení s názvem *Server* a *Client* a vložíme ho na pracovní plochu simulátoru. Poté klikneme na ikonu směrovače, vybereme *VyOS* a vložíme ho na plochu dvakrát. Nakonec v levém

menu vybereme ikonu přepínače, vybereme přepínač *Ethernet switch* a vložíme ho na plochu také dvakrát. Na obrázku 7-1 jsou vysvětlivky ikon, které najdeme v levé liště.



Obr. 7-1: Význam ikon v levé liště

Potom směrovače propojíme – spojíme port *eth0* na směrovači *R1* s portem *eth0* na směrovači *R2*. Dále propojíme směrovače s přepínači a to tak, že na směrovači *R1* připojíme port *eth1* k přepínači *Switch1* na libovolný port a na směrovači *R2* připojíme port *eth1* k přepínači *Switch2* na libovolný port. Nakonec ještě připojíme zařízení *Server-1* port *Ethernet0* k přepínači *Switch1* a zařízení *Client*-1 port *Ethernet0* připojíme k přepínači *Switch2*. Topologie by měla vypadat jako na obrázku 7-2.



Obr. 7-2: Topologie sítě

Teď, když máme topologii zapojenou můžeme zařízení zapnout pomocí zeleného trojúhelníku v horní liště. Když se zařízení zapnou, tak je potřebné nejdřív nakonfigurovat směrovače, aby mohli mezi sebou VM komunikovat. Do zařízení se dostaneme dvojklikem na konkrétní prvek. Takže si zapneme konzoli směrovače R1 a přihlásíme se do něj pomocí jména *student* a hesla *student*, stejně tak do směrovače R2. Pote si otevřeme textový soubor R1.txt a R2.txt zkopírujeme jeho obsah a vložíme pravým klikem na myši

do konzole směrovačů. Po těchto příkazech by měli VM být schopné komunikovat. Do VM přihlásit pomocí jména *student* a hesla *student.* Do VM se dostaneme dvojklikem na daný prvek v síti (server, client). Při práci s VM je doporučeno mít zapnutou anglickou klavesnici. Na VM by měly být nastavené IP adresy jako na obrázku 5-2. To můžeme ověřit pomocí příkazu *ip address show* – příkaz vypíše informace o všech rozhraních, které ve VM máme. Pokud chceme, aby nám ukázal jen IP adresu ethernetového rozhraní (operační systém přidělil rozhraní název ens3), zadáme následující příkaz:

$ ip address show ens3 | grep inet

Příkaz nám vypíše IP adresy, které jsou na konkrétním rozhraní. Že VM spolu dokážou komunikovat, můžeme vyzkoušet pomocí pingu. Když nekomunikují, ověřte, jestli jste nezaměnili konfigurace směrovačů, popřípadě nenakonfigurovali oba směrovače stejnou konfigurací a zkuste to opravit.

Teď, když víme, že VM spolu dokážou komunikovat, můžeme pokračovat dále. Na to, abychom mohli porovnat TCP a UDP, budeme potřebovat na serveru spustit HTTP a TFTP server. Na serveru a klientovi je nainstalovaný všechen potřebný software, je potřebné ho jen spustit a nastavit firewall, aby mohla komunikace probíhat.

Začneme s konfigurací serveru. První krok bude spustit na serveru HTTP a TFTP server a nastavit ho tak, aby se spouštěl i při restartování VM, to uděláme následujícími příkazy:

$ sudo systemctl start httpd // zapnutí HTTP serveru
$ sudo systemctl enable httpd // povolení zapnutí HTTP serveru při startu
$ sudo systemctl start tftp // zapnutí TFTP serveru
$ sudo systemctl enable tftp // povolení zapnutí TFTP serveru při startu

Příkaz *sudo* slouží k udělení práv správce pro použitý příkaz a bude potřebné zadat heslo, které je „student“. Když už máme spuštěné aplikace pro HTTP a TFTP server, je ještě potřebné nastavit firewall na straně serveru, aby byla komunikace možná. CentOS má jako předinstalovaný firewall *firewalld* (Dynamic Firewall) a pomocí jednoduchých příkazů je možné do firewallu přidat služby, které chceme využívat:

$ sudo firewall-cmd --permanent --add-service=http
$ sudo firewall-cmd --permanent --add-service=tftp

Atribut *--permanent* znamená, že změna, kterou chceme ve firewallu udělat, tak bude přidána nastálo, ale až po restartu firewallu. Že tam teď není, můžeme ověřit příkazem *$ sudo firewall-cmd --list-all*. Firewall restartujeme příkazem:

$ sudo firewall-cmd --reload

Nyní, když máme zapnutý aplikační server HTTP a TFTP, už jen potřebujeme vytvořit nějaký soubor, který budeme chtít stáhnout ze serveru pomocí klienta. V případě HTTP serveru je potřebné vytvořit soubor ve složce, která se nachází */var/www/html/*. Do složky se dostaneme pomocí příkazu cd:

$ cd /var/www/html/ // změna pracovní složky

Ve složce si vytvoříme pomocí textového editoru vim (popřípadě je možné využít i editor nano) nový soubor následovně:

$ sudo vim fileHTTP // vytvoří soubor fileHTTP

Otevře se nám ne moc přátelský editor textu. Pro zadání textu stiskneme na klávesnici klávesu *i*, dále můžeme zadat text „Hello World“. Pro ukončení editování je potřebné na klávesnici stisknout klávesu *Esc*. Pokud chceme soubor zapsat a vypnout editor, zadáme příkaz *:wq, :* (dvojtečka) slouží pro zadávání příkazu, *w* (write) je příkaz pro zapsání, *q* (quit) je příkaz pro vypnutí. Teď, když máme vytvořený soubor, chceme stejný soubor přenést i přes TFTP, abychom mohli pozorovat rozdíly. Soubor už nemusíme vytvářet,

stačí, když ho zkopírujeme do složky pro TFTP server a ta je */var/lib/tftpboot/*. Kopírování uděláme následujícím příkazem:

$ sudo cp fileHTTP /var/lib/tftpboot/fileTFTP // zkopíruje soubor

Teď máme připravené jednoduché soubory pro přenos a můžeme se přesunout na klienta. Pro HTTP přenos nepotřebujeme na klientovi nic nastavovat, ale pro TFTP přenos je potřebné na klientovi povolit ve firewalle příchozí přenos od serveru. Uděláme to tak, že přidáme do firewallu IP adresu serveru a povolíme příchozí připojení:

$ sudo firewall-cmd --permanent --zone=trusted --add-source=192.168.10.11/24

Po tomto příkazu je potřebné jen restartovat firewall stejně jako na serveru:

$ sudo firewall-cmd --reload

Ještě před přenosem si vytvoříme na klientovi složku *files*, kam budeme stahovat soubory pomocí příkazu *mkdir files* a potom *cd files*. Teď, když máme všechno připraveno, si v GNS3 vybereme jednu z dvou linek, které jsou mezi serverem a směrovačem R1. To uděláme tak, že na linku klikneme pravým tlačítkem myši a vybereme *Start capture*, tím se otevře malé okno a tam vybereme *Ok*. Tímto se nám spustí Wireshark, který bude na vybrané lince odchytávat pakety. Pro stažení souboru ze serveru pomocí HTTP protokolu použijeme program *wget*:

$ wget http://192.168.10.11/fileHTTP // příkaz zadán na klientovi

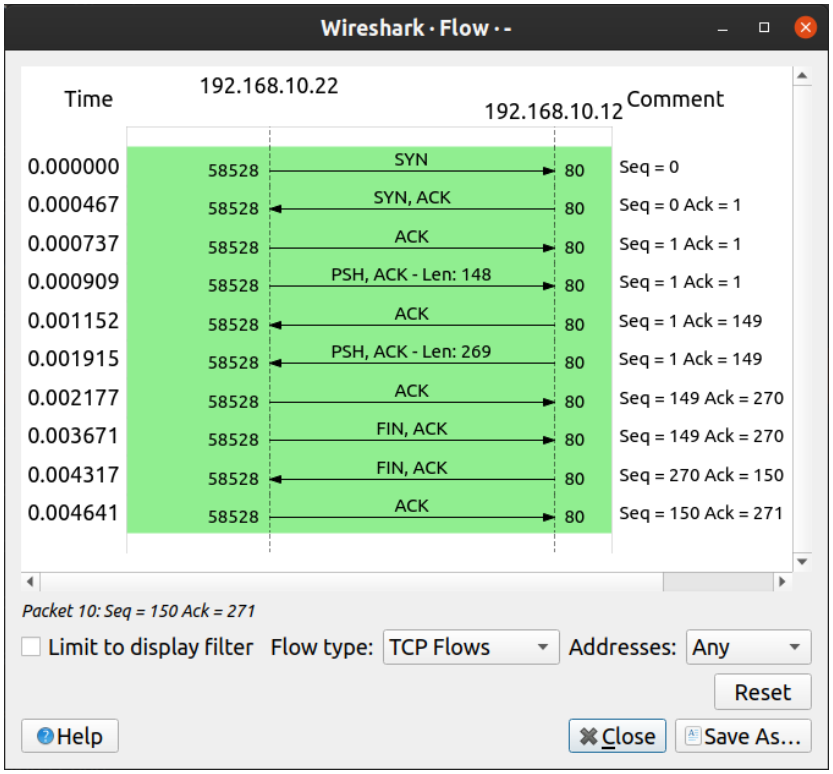
Průběh přenosu můžeme sledovat pomocí procent, které nám ukazuje přímo program *wget*. Tímto se nám stáhne ze serveru soubor *fileHTTP. Ž*e je to ten stejný soubor, který jsme vytvořili, můžeme ověřit tak, že ho buď otevřeme pomocí *v*im a podíváme se na obsah, nebo si vypíšeme jeho obsah s příkazem *cat:*

$ cat fileHTTP

Příkaz by měl vypsat obsah souboru – Hello *World*. Teď přeneseme soubor pomocí *TFTP* a to pomocí příkazu:

$ tftp -v 192.168.10.11 -c get fileTFTP

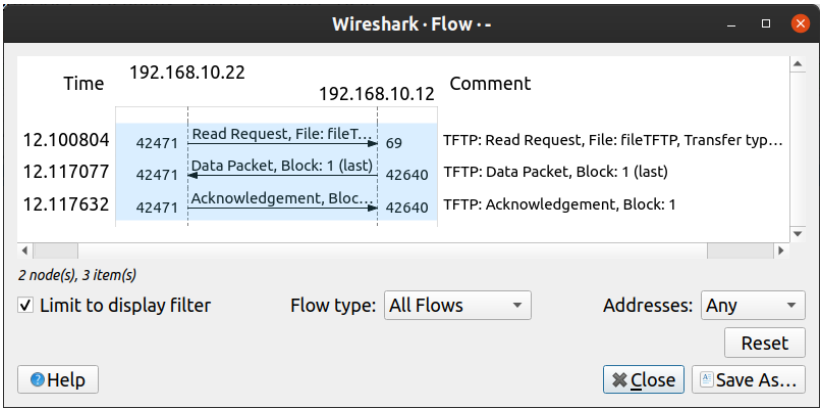
Průběh přenosu není vidět, ale to, že přenos skončil, poznáme tak, že můžeme opět zadávat příkazy do konzole. Pomocí příkazu *ls* nebo *ll* si můžeme vypsat obsah složky. Měly by tam být dva soubory – *fileHTTP* a *fileTFTP*. Pomocí příkazu *cat* můžeme zkontrolovat, že soubory obsahují stejný obsah. Teď můžeme na lince v *GNS3* vypnout zachytávání paketů. Klikneme na linku pravým tlačítkem a vybereme *Stop capture*. Potom si otevřeme spuštěný *Wireshark* a měli bychom mít zachycený TCP i UDP přenos. Na to, abychom se podívali, jak vypadá TCP přenos podrobněji, klikneme v horní liště na *Statistics* a vybereme *Flow Graph*. Otevře se nám okno a v jeho spodní části vybereme *Flow type – TCP Flows*. Ten by měl vypadat podobně jako na obrázku 7-3.



Obr. 7-3: Flow graf TCP přenosu

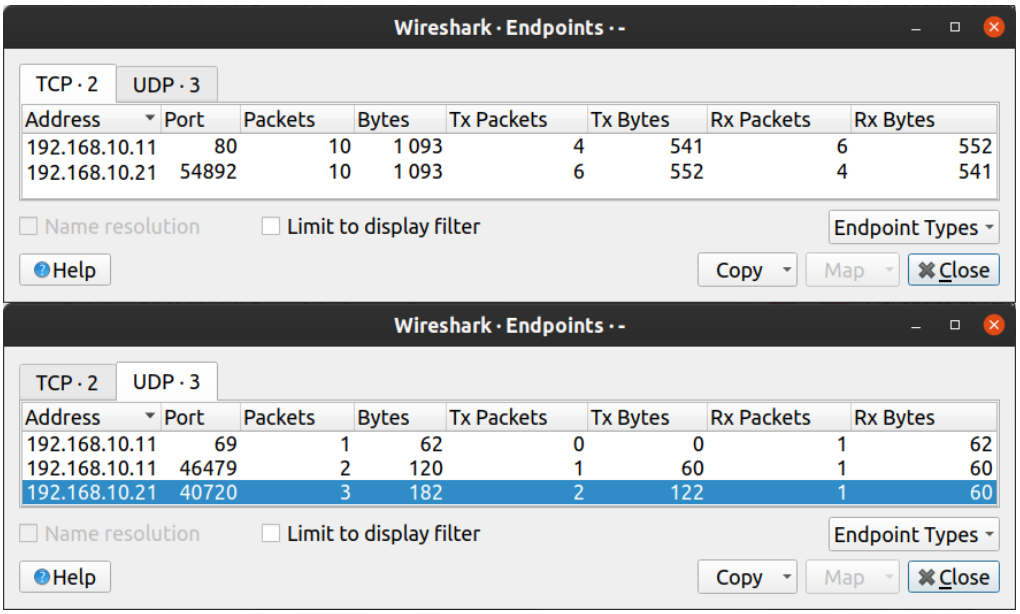
Jak můžeme vidět, tak na přenos jediného souboru s textem *Hello World* je potřebných 10 paketů. Přenos začíná ze strany klienta na port 80 na serveru, kam klient posílá *TCP* segment s nastaveným SYN a sekvenčním číslem 0. Server mu odpoví *TCP* paketem se *SYN* a *ACK*. Můžeme sledovat, jak se postupně zvětšuje sekvenční číslo (*SEQ*) a potvrzovací číslo (*ACK*). Číslo *ACK* slouží na potvrzení a zároveň je to číslo, které odesílatel očekává jako další *SEQ* číslo od příjemce. Komunikace se ukončuje nastaveným bitem *FIN*, jak můžeme vidět na obrázku 7-3. Nejdřív klient odešle segment s nastaveným *FIN* a poté mu server odpoví s nastaveným bitem *FIN*. Komunikace se ukončí, když klient odešle poslední segment s nastaveným *ACK*, že přijal *FIN* od serveru. Při TCP přenosu se pořád posílají pakety s nastaveným příznakovým bitem *ACK* a to proto, že když se posílá bez něj, tak se pořadové číslo *ACK* ignoruje a zbytečně se využívá přenosové pásmo. Tudíž se bit *ACK* přenáší pořád kromě prvního paketu, kdy odesílatel neví, jaké pořadové číslo tam má být.

Teď se můžeme podívat na to, jak probíhal přenos pomocí TFTP, který běží na UDP protokolu. Půjdeme do hlavního okna *Wireshark* a nastavíme si filtr na *udp*, potom se vrátíme do flow grafu a zaklikneme políčko *Limit to display filter*, tím nám tam zůstane pouze UDP přenos, který by měl vypadat podobně jako na obrázku 7-4.



Obr. 7-4: UDP flow graf

Jak je vidět podle obrázku, tak na přenos souboru pomocí UDP stačí jenom tři pakety. Přenos začíná na straně klienta požadavkem na přečtení souboru, následně mu server pošle paket s daty, který je zároveň označený jako poslední, a nakonec mu klient jen odešle potvrzovací paket. Potvrzovací paket *Acknowledgement* není součástí protokolu UDP, jelikož UDP nemá mechanizmus na kontrolu, jestli druhá strana přijala paket, proto to musí být vyřešené na aplikační vrstvě, aby bylo zaručené, že klient obdržel všechna data. TFTP server neodesílá další pakety, dokud od klienta nedostane potvrzení o přijetí posledního odeslaného paketu. Jak vidíme z *Wireshark*, tak na přenos malého souboru je potřeba přenést víc dat při přenosu pomocí protokolu *TCP*. Kolik dat se přesně přeneslo přes TCP a UDP, je možné ověřit ve *Wireshark*. V horní liště vybereme *Statistics* – *Endpoints*, otevře se nám okno, které nám ukazuje, kolik dat se přeneslo do konkrétních koncových bodů. Měly by tam být záložky *TCP* a *UDP*, pokud tam nejsou, je potřeba je vybrat v *Endpoint Types*. Když záložky otevřeme, vidíme, že přes *TCP* se přeneslo 1093 bajtů a přes UDP se přeneslo jen 182 bajtů, jak můžeme vidět na obrázku 7-5. To je množství dat konkrétně pro náš soubor s textem *Hello World*. Kdyby se přenášel jiný soubor, bude množství dat jiné.



Obr. 7-5: Objem přenesených dat

Po získání všech grafů si zachycené pakety uložíme pomocí File – Save as. Wireshark soubory si budeme ukládat i v ostatních scénářích, abychom je pak mohli prezentovat vyučujícímu.

7.3 Výpadek během přenosu TCP a UDP

V následující části úlohy budeme simulovat a sledovat, jak se chovají protokoly UDP a TCP, když během přenosu nastane výpadek v síti. Budeme používat stejné VM jako v předchozím scénáři. Na přenos TCP opět použijeme HTTP server, pro UDP přenos je ve VM vytvořen jednoduchý skript v jazyce Python. Klient odešle paket na server a server mu začne odesílat pakety, dokud nevypneme program. Na straně serveru není žádný mechanizmus, který by kontroloval, jestli klient paket přijal, nebo ne. Pro povolení komunikace UDP mezi klientem a serverem je potřebné na serveru povolit port 20001/*UDP;* to se povolí ve firewallu pomocí následujícího příkazu:

$ sudo firewall-cmd --permanent --add-port=20001/udp

Následně je potřebné firewall znovu načítat:

$ sudo firewall-cmd --reload

Topologie zůstane stejná jako v přecházejícím scénáři z obrázku 5-2.

Ještě před začátkem TCP přenosu zapneme zachytávání paketů mezi serverem a přepínačem 1 a mezi klientem a přepínačem 2, budeme sledovat obě strany, co se děje během výpadku. Výpadek nastane tak, že na směrovači *R2* vypneme rozhraní sériové linky *eth0*. Pro přenos budeme používat soubor o velikosti 250 kB a vytvoříme si i soubor o velikosti 500 kB pro další scénář. Přenos trvá přibližně 8 vteřin.

$ cd /var/www/html

$ sudo truncate -s 250k file250k

$ sudo truncate -s 500k file500k

Ještě před začátkem přenosu v směrovači R2 nastavíme příkaz pro vypnutí rozhraní. Potřebujeme dostat do režimu pro konfiguraci rozhraní *eth0*, takže nejdřív musíme vejít do konfiguračního režimu pomocí příkazu:

student@student:~$ configure

Zadáme příkaz pro vypnutí rozhraní *eth0*:

student@student~$ set interfaces ethernet eth0 disable

A pro vypnutí rozhraní si připravíme příkaz *commit*, který nespustíme do doby, než bude probíhat přenos mezi serverem a klientem:

student@student~$ commit

Teď, když máme zapnuté zachytávání paketů a připravený příkaz pro vypnutí rozhraní na směrovači, můžeme spustit stahování na straně klienta pomocí příkazu:

$ wget http://192.168.10.11/file250k

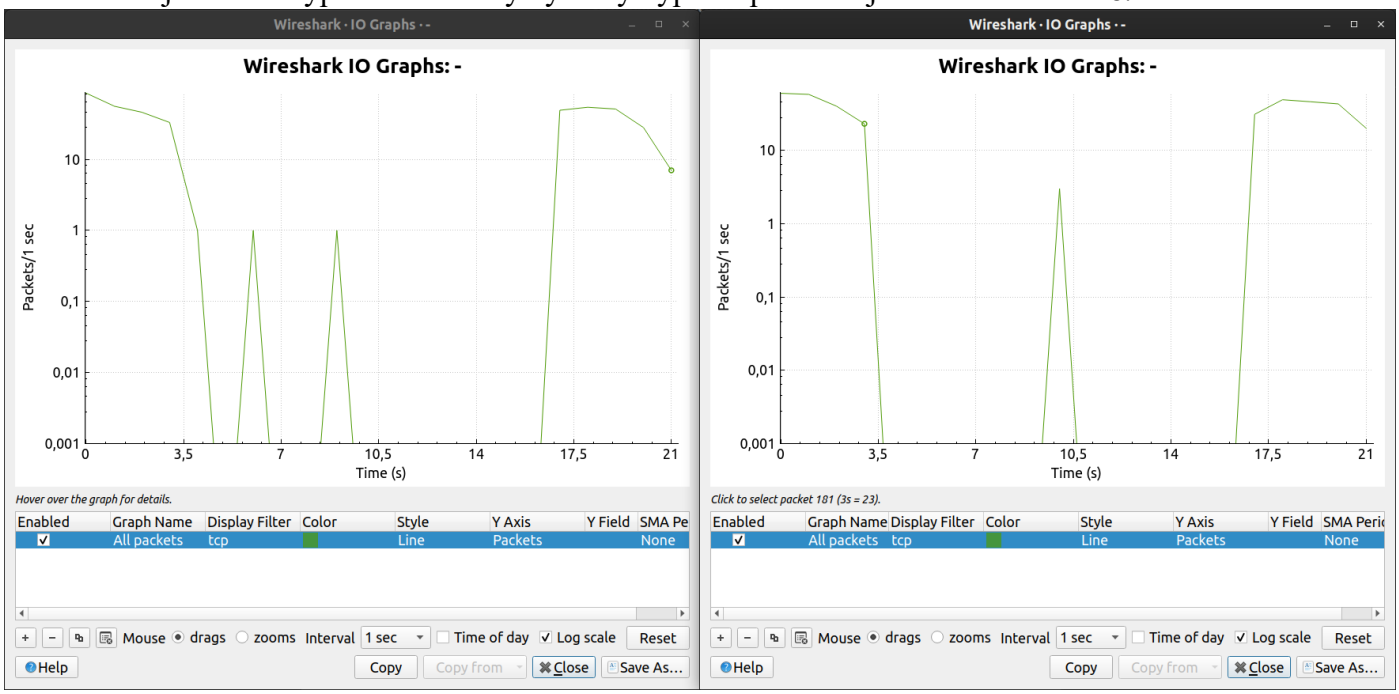
V době, kdy se soubor stahuje, odešleme příkaz směrovači *R2,* aby se port *eth0* vypnul. Po pár vteřinách zapneme rozhraní eth0 na směrovači pomocí těchto dvou příkazů:

student@student~$ delete interfaces ethernet eth0 disable

student@student~$ commit

Když stahování skončí, vypneme v GNS3 zachytávání paketů a otevřeme si obě *Wireshark* okna, která máme spuštěná. Měli bychom v nich mít zachycené pakety z přenosu. V obou z nich si otevřeme *I/O Graph* jako v předchozím scénáři, nastavíme si ve spodním filtru *tcp* a nastavíme interval. Necháme na jedné vteřině a zapneme si

logaritmické měřítko tím, že zaškrtneme políčko *log scale*, abychom lépe viděli, co se děje během výpadku. Průběhy by měly vypadat podobně jako na obrázku 7-6.



Obr. 7-6: Výpadek během TCP přenosu

Na obrázku vlevo je zachycený přenos mezi serverem a přepínačem 1 a na pravé straně je zachycený přenos mezi přepínačem 2 a klientem. Jak je podle obrázku zřejmé, tak výpadek zaznamenaly obě strany a během výpadku se poslalo jen několik paketů. Jak můžeme vidět, v tomto případě se na straně serveru poslaly pouze dva pakety v tomto konkrétním případě. Když na ně klikneme a podíváme se do *Wireshark*, tak zjistíme, že to jsou pakety se stejným sekvenčním číslem, jako má poslední odeslaný paket. Jelikož server neobdržel od klienta potvrzovací paket, snaží se odeslat paket znovu a vyjde mu to až na třetí pokus.

Teď provedeme UDP přenos a budeme sledovat, co se při něm děje, když nastane výpadek. Skripty jsou uložené ve VM v serveru i na klientovi ve složce */home/student/scripts/*. Na obou se do složky dostaneme pomocí příkazu cd. Když už jsme ve složce, tak si pomocí příkazu *ll* zobrazíme soubory, které ve složce jsou. Na serveru by měl být soubor udp_server.py a na klientovi by měl být soubor udp_client.py. Na serveru můžeme skript spustit pomocí příkazu:

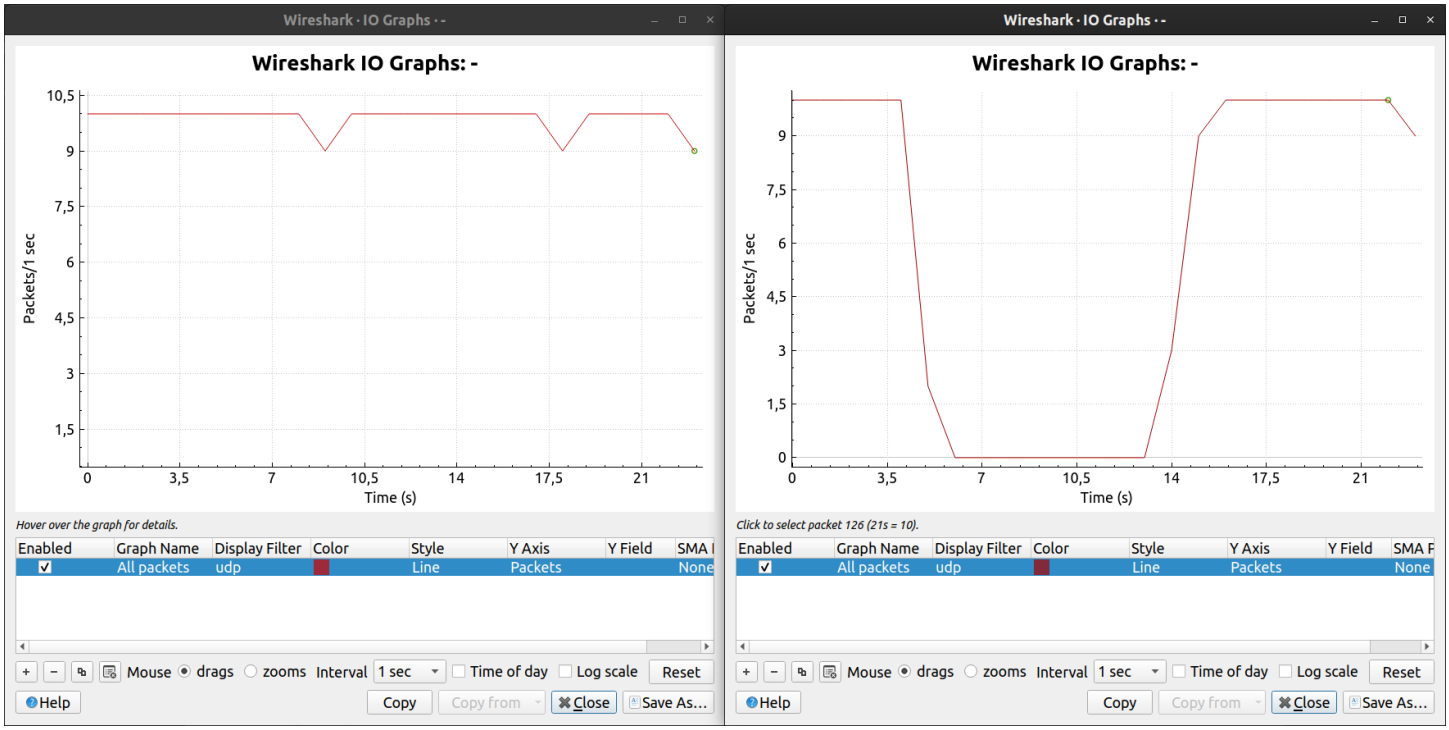
$ python2 udp_server.py

V konzoli se nám vypíše, že server je spuštěn. Když spustíme skript na straně klienta, můžeme vidět, že mezi klientem a serverem probíhá komunikace. To vidíme tak, že na straně serveru i na straně klienta se nám zvyšuje číslo poslaných a přijatých paketů. Skript na straně klienta spustíme pomocí následujícího příkazu:

$ python2 udp_client.py

Skripty vypneme pomocí zkratky Ctrl+C, nejdřív na straně serveru a potom na straně klienta. Číslo odeslaných a přijatých paketů vypsané pomocí skriptu by mělo být stejné. Výpadek budeme simulovat stejně jako v případě TCP přenosu a to vypnutím ethernetové linky na směrovači R2 podle již známých příkazu a zapneme ji zpátky pomocí známých příkazů. Teď uděláme to stejné jako při TCP přenosu – zapneme zachytávání paketů mezi serverem a přepínačem 1 a klientem a přepínačem 2. Nejdřív spustíme skript na straně

serveru a potom na straně klienta. Měla by začít komunikace, během které vypneme ethernetovou linku na R2 a po pár vteřinách ji znovu zapneme, potom vypneme skript nejdřív na straně serveru a potom na straně klienta. Počty paketů vypsané v konzoli by neměly být stejné, na straně serveru by mělo být vyšší číslo. Potom vypneme zachytávání paketů v GNS3 a zapneme si I/O graf na obou stranách jako na obrázku 7-7.



Obr. 7-7: Výpadek během UDP přenosu

Na obrázku 7-7 je na levé straně průběh přenosu mezi serverem a přepínačem 1 a na pravé straně je průběh přenosu mezi přepínačem 2 a klientem. Jak můžeme vidět, v době, kdy nastal výpadek, server dále posílal pakety a nezajímal se o to, jestli klient pakety přijímá. Server na své straně počítá, kolik paketů odeslal, a klient na své straně počítá, kolik paketů od serveru přijal. Ve výpisu konzole můžeme po ukončení programu vidět, že počet serverem odeslaných paketů je větší než počet paketů přijatých klientem.

7.4 Zahazování paketů

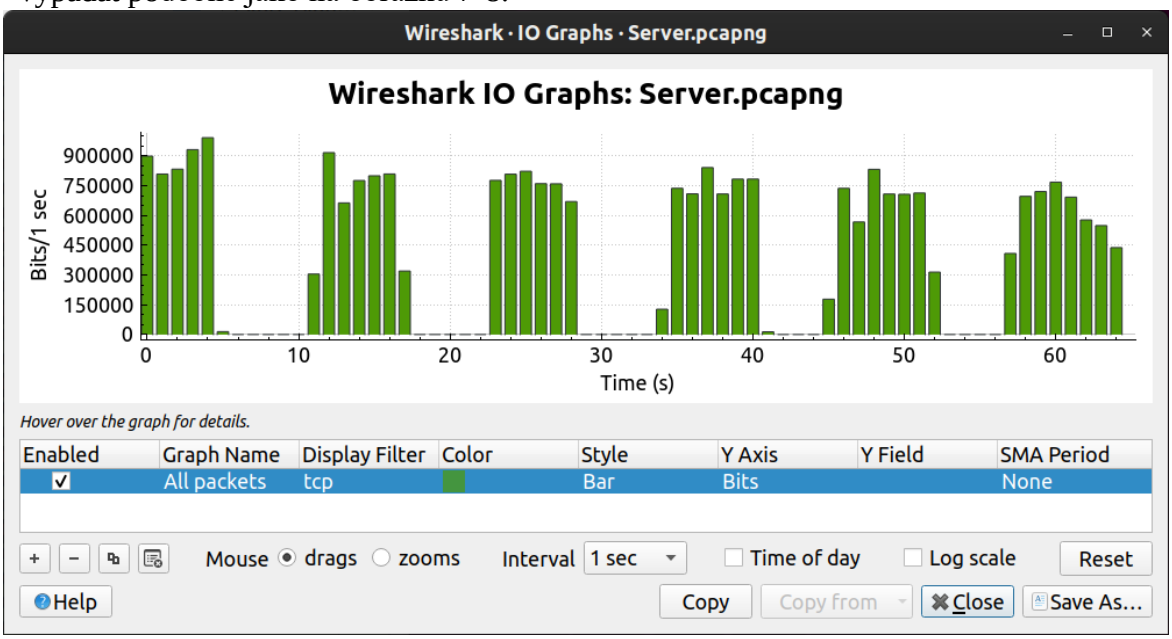
V této části si ukážeme, jak se protokoly chovají, když nenastane výpadek, ale občas se některý paket ztratí nebo zahodí. Pro zahazování využijeme filtr, který je dostupný v simulátoru *GNS3*. Je možné v něm nastavit frekvenci zahazování paketů, pravděpodobnost zahazování paketů a další věci. Pro TCP přenos využijeme stejně jako v přecházejících případech HTTP přenos a pro UDP přenos stejný skript jako v předešlém případě. Do topologie nepotřebujeme nic přidávat, takže jen na VM klienta změníme aktuální složku na */home/student/files/* pomocí příkazu *cd*. Ve složce vymažeme všechny soubory pomocí příkazu *rm -rf **, pokud tam nějaké jsou. Budeme přenášet soubor o velikosti 500 kB, který jsme si vytvořili v předcházejících krocích.

Potom si v *GNS3* zapneme zachytávání paketů na lince mezi serverem a přepínačem 1 klikneme pravým tlačítkem myši na linku mezi směrovačem *R1* a *R2* a vybereme *Packet filters*. Poté si otevřeme záložku *Packet loss*, ve které je možné nastavovat zahazování

paketů v procentech. Budeme simulovat zahazování paketů od 0% až do 5%. Pro každé zahazování paketů budeme přenášet stejný soubor o velikosti 500 kB. Budeme postupovat tak, že nejdřív dáme stahovat soubor pomocí následujícího příkazu:

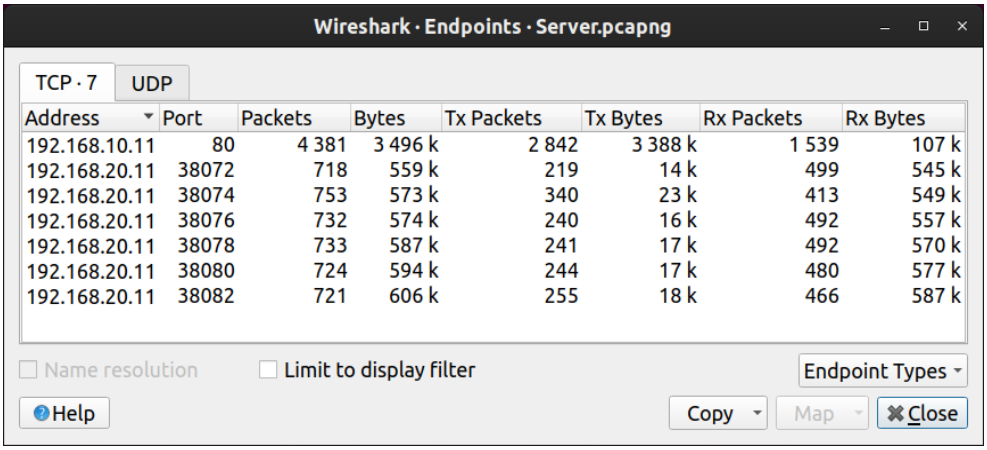
$ wget http://192.168.10.11/file500k

Při prvním stahování budeme mít zahazování nastavené na 0% a postupně ho budeme zvětšovat o jedno procento až do 5%. To provedeme tak, že nejdřív nastavíme zahazování a potom dáme stahovat soubor pomocí *wget*. Až budeme mít všechny přenosy ukončené, pomocí tlačítka *Reset* vypneme všechny nastavení ve filtru, okno i zachytávání paketů. Přejdeme do *Wireshark* a otevřeme si *I/O* graf. Filtr si nastavíme na *tcp,* styl grafu nastavíme na *bar* a na y os si nastavíme zobrazování v bitech za vteřinu. Graf by měl vypadat podobně jako na obrázku 7-8.



Obr. 7-8: Zahazování paketu během TCP přenosu

Na obrázku můžeme vidět graf toho, jak vypadaly jednotlivé přenosy a že rychlost přenosu se navzdory zahazování výrazně neměnila. Více by nás zajímalo, jaké množství dat se při jednotlivých přenosech přeneslo, protože při větším zahazování se musí teoreticky přenést větší množství dat z důvodu toho, že se ztracená data musí přeposílat. Takže si ve Wireshark otevřeme *Statistics – Endpoints,* klikneme na záložku TCP a okno *Endpoints* by mělo vypadat jako na obrázku 7-9.



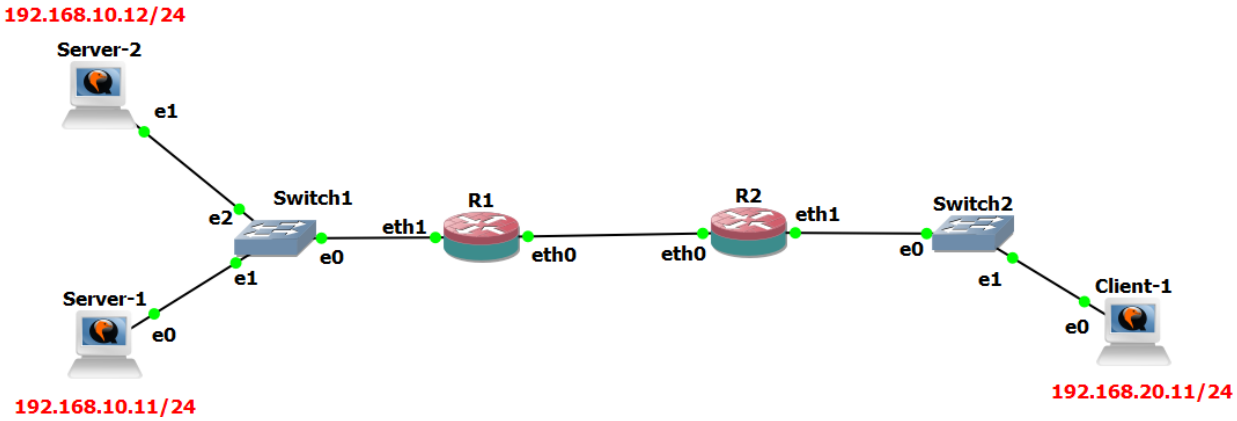
Obr. 7-9: Množství přenesených dat při zahazování

V okně můžeme vidět, kolik dat se přeneslo při jednotlivých nastaveních zahazování. Pokaždé, když se program *wget* spustí, se jeho port zvýší o 2. V tomto případě při prvním přenosu, kde bylo zahazování nastaveno na 0 %, byl port 38072, při 1 % byl port 38074, a tak se to zvyšuje až po 5%. Ve sloupci *Bytes* můžeme vidět množství přenesených dat, při 0 % se přeneslo 559 kB, při 1% se přeneslo 573 kB a dále se s větším zahazováním zvyšuje i množství přenesených dat, kde se při 5% přeneslo až 606 kB.

7.5 Zahlcení TCP přenosu

V této části laboratorní úlohy budeme simulovat zahlcení linky během *TCP* přenosu a sledovat, jak se bude *TCP* přenos měnit. Na *TCP* přenos bude opět použitý *HTTP* protokol, pomocí kterého budeme stahovat ze serveru soubor. Na zahlcení použijeme další VM, která má v sobě skript, který odesílá velké množství *UDP* paketů v krátkém čase na IP adresu, kterou směrovač *R2* zahazuje. Při tomto přenose zahlcujeme linku mezi směrovačem *R1* a *R2*.

Do topologie přidáme další server z levé lišty, uložíme ho nad server jedna, připojíme ho do stejného přepínače jako Server-1, ale připojíme ho přes rozhraní Ethernet1 do libovolného portu přepínače 1. Topologie by měla vypadat jako na obrázku 7-10.



Obr. 7-10: Topologie zapojení

Zařízení mají nastavené IP adresy podle obrázku. Simulace zahlcení bude probíhat tak, že klient požádá server o stažení souboru pomocí HTTP protokolu a během toho spustíme

na serveru 2 skript, který odešle určité množství paketů na IP adresu *192.168.30.1*, na kterou má směrovač *R1* nastavenou statickou cestu, aby tyto data odesílal na směrovač *R2*. Směrovač *R2* má nastavené, aby pakety, které směrují do sítě 192.168.30.0/24, zahazoval a neodesílal zpátky *ICMP* zprávy o nedostupnosti.

Když máme přidaný server 2, můžeme ho spustit tím, že na něj klikneme pomocí pravého tlačítka myši, vybereme *Start* a počkáme, až se zařízení zapne. Přihlásíme se do něj pomocí jména *student* a hesla *student*. Pomocí následujícího příkazu ověříme, že má zařízení správnou IP adresu a to 192.168.10.12:

$ ip address show ens4 | grep inet

Pokud má správnou IP adresu, můžeme pomocí pingu ověřit, jestli dokáže komunikovat s ostatními zařízeními. Na serveru přejdeme do složky */home/student/scripts* pomocí příkazu *cd* a pomocí příkazu *ll* si vypíšeme obsah složky. Měly by tam být dva skripty, jeden z nich jsme již využívali a teď budeme využívat ten druhý. Skript *udp_sender.sh* je napsaný v jazyce *bash* a jediné, co dělá, je, že odešle určitý počet paketů, které obsahují zprávu *Hi* a které směrovač *R2* zahodí, když k němu dojdou, na adresu 192.168.30.1. Skript má jeden vstupní parametr a to je kolik paketů chceme odeslat. Pro náš případ budeme odesílat 500, 1000, 3000, 6000 a 10000 paketů a budeme sledovat, co se bude dít s *TCP* přenosem, když sériovou linku takto zahltíme.

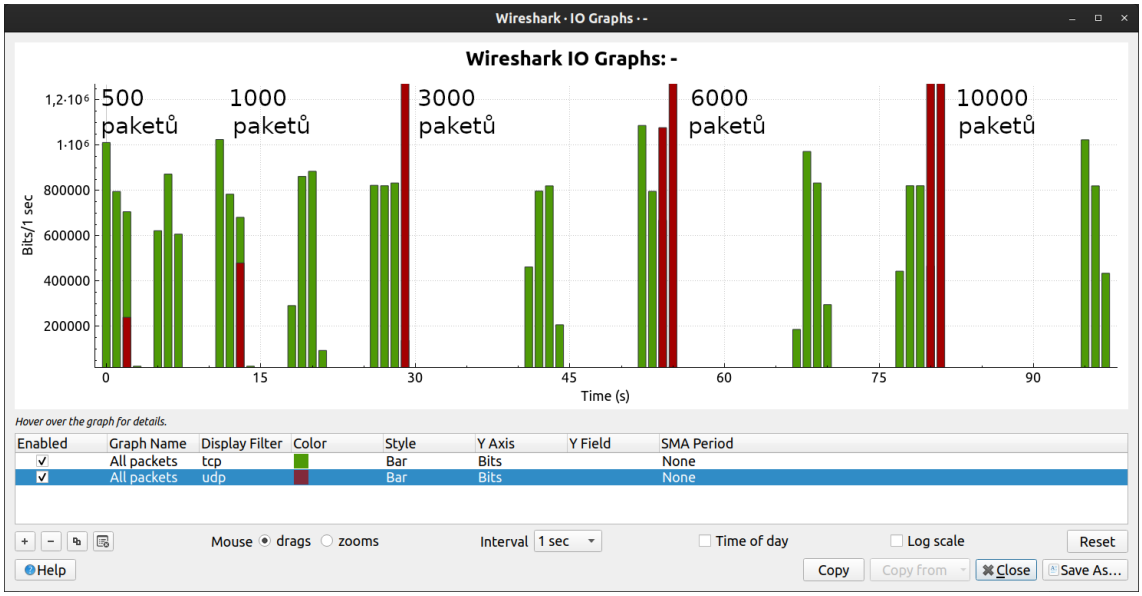
Takže si na klientovi připravíme příkaz pro stahování souboru o file500k, který jsme používali i v předcházejících scénářích:

$ wget http://192.168.10.11/file500k

Na serveru 2 si připravíme příkaz pro spuštění skriptu:

$ ./udp_sender.sh 500

Číslo 500 je vstupní proměnná pro skript, která určuje, kolik paketů se má odeslat. Teď si zapneme zachytávání paketů mezi směrovačem *R1* a přepínačem *Switch1*. Potom spustíme stahování souboru na klientovi a během stahování spustíme skript na serveru 2. Když stahování skončí, dáme stahovat znova a opět spustíme skript, ale už ne se vstupním parametrem 500 ale 1000, a takhle pokračujeme dále pro 3000, 6000 a 10000. Když máme všechny přenosy ukončené a pakety zachycené, můžeme v *GNS3* vypnout zachytávání paketů a otevřít si *Wireshark* se zachycenými pakety. Otevřeme si *I/O* graf a nastavíme si první filtr na *tcp*, styl grafu vybereme bar a y os nastavíme na bity za vteřinu. Druhý filtr nastavíme na udp, stejný styl grafu bar a y os znova nastavíme na bity za vteřinu. Graf by měl vypadat podobně jako na obrázku 7-11.



Obr. 7-11: Zahlcení TCP přenosu

Jak můžeme vidět podle grafu, čím více jsme linku zahltili pakety, tím déle TCP přenos neprobíhal, avšak byl schopný se poté obnovit. Rychlost linky mezi směrovači je nastavená na 700000 bitů za vteřinu, jak jsme ji nastavili v předcházejících krocích. Jak můžeme vidět, tak UDP pakety tuto hodnotu přesáhly při odesílání 3000 a více paketů a TCP přenos se na nějakou dobu zastavil.

7.6 Kontrolní otázky

1. Proč se při přenosu UDP pomocí protokolu TFTP posílají potvrzovací pakety Acknowledgment?
2. Když nastal výpadek, tak se komunikace při protokolu TCP zastaví. Jak server pozná, že nastal výpadek a přestal posílat data klientovi? A když jsme linku obnovili, jak server zjistil, že může data klientovi dále posílat?
3. Proč při výpadku během UDP komunikace server dále klientovi posílal data, i když klient nebyl dostupný?
4. Proč se protokol UDP využívá, když je nespolehliví?
5. Jaké výhody má UDP protokol na rozdíl od protokolu TCP a naopak?
6. Pro jaké služby se protokoly využívají?

8. LABORATORNÍ ÚLOHA – SMĚROVACÍ PROTOKOLY

8.1 Teoretický úvod

Směrovací protokoly slouží k tomu, aby směrovače věděly, kam mají poslat paket tak, aby se dostal ke správnému cíli. Pomocí směrovacích protokolů se směrovače umí dorozumět a předávat si informace o známých sítích. Internet by mohl fungovat i bez nich, protože je teoreticky možné na všechny směrovače nakonfigurovat statické směrování, ale když vezmeme v úvahu, jak velká síť nternet je, tak prakticky je to nemožné. Dále pak statické směrování nedokáže reagovat na změny v síti, kdežto směrovací protokoly to dokážou. Mezi nejznámější směrovací protokoly patří například OSPF (Open Shortest Path First), EIGRP (Enhanced Interior Gateway Routing Protocol), RIP (Routing Information Protocol) a další. Směrovací protokoly se dělí na směrovací protokoly uvnitř autonomního systému a směrovací protokoly mezi autonomními systémy. Do první skupiny patří například OSPF, EIGRP a RIP, do druhé skupiny patří například EGP (Exterior Gateway Protocol), BGP (Border Gateway Protocol).

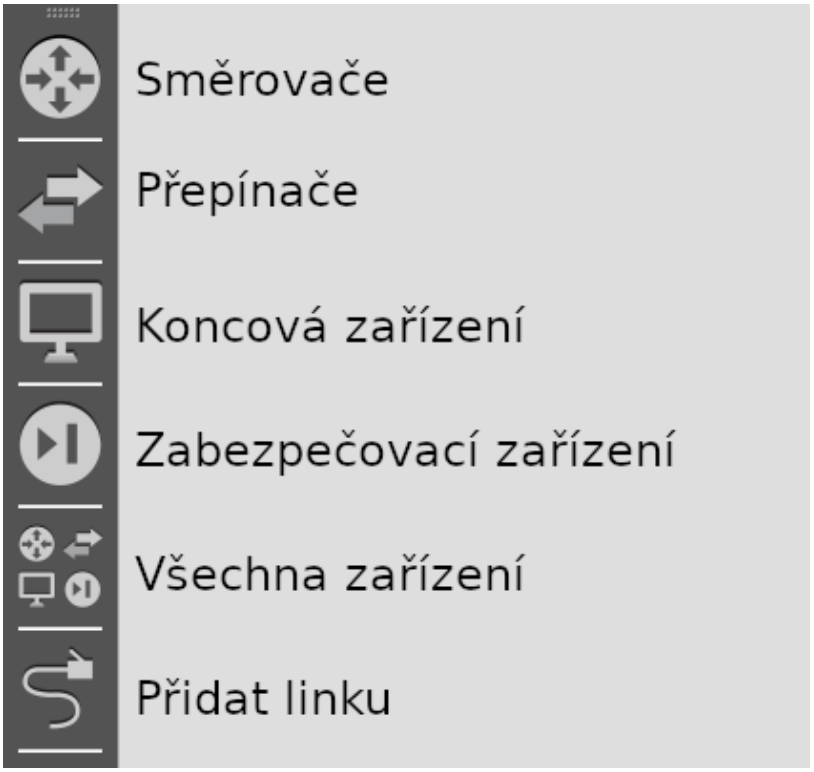
V této laboratorní úloze si ukážeme rozdíly mezi směrovacími protokoly OSPF a RIP verze 2 a některé základní funkcionality protokolu OSPF. Na simulaci scénáře využijeme simulační program GNS3 a na simulaci směrovačů využijeme volně dostupný operační systém VyOS, který podporuje OSPF a RIP ve verzi 2.

8.2 Základní srovnání OSPF a RIP

Protokoly OSPF a RIP jsou směrovací protokoly. Mezi těmito protokoly je mnoho rozdílů. Protokol RIP je jednodušší než OSPF. RIP je takzvaný *distance vector protocol* a OSPF je takzvaný *link state protocol*, to znamená, že RIP vybírá nejkratší cestu podle počtu skoků a protokol OSPF vypočítává nejkratší cestu podle rychlostí a ceny linky. Konkrétně protokol RIP si vybírá logicky nejkratší možnou cestu, protokol OSPF počítá nejkratší cestu pomocí Dijkstrova algoritmu. V případě protokolu RIP je maximální počet skoků nastaven na 15, v případě OSPF je počet skoků neomezen. Další rozdíl je v tom, jak se v RIP a OSPF přeposílají změny v síti. Směrovače v RIP posílají pokaždé všechny informace ze své tabulky, a to v intervalu 30 sekund, kdežto v případě protokolu OSPF se posílají pouze změny v síti, a tak je konvergence v případě OSPF rychlejší, než u protokolu RIP. Rozdíl je ještě v tom, že protokol OSPF může svou síť rozdělit na oblasti a v oblastech se můžou sumarizovat sítě, a proto směrovače v jiných oblastech vidí pouze jednu sumarizovanou síť místo všech sítí, které se v oblasti nacházejí.

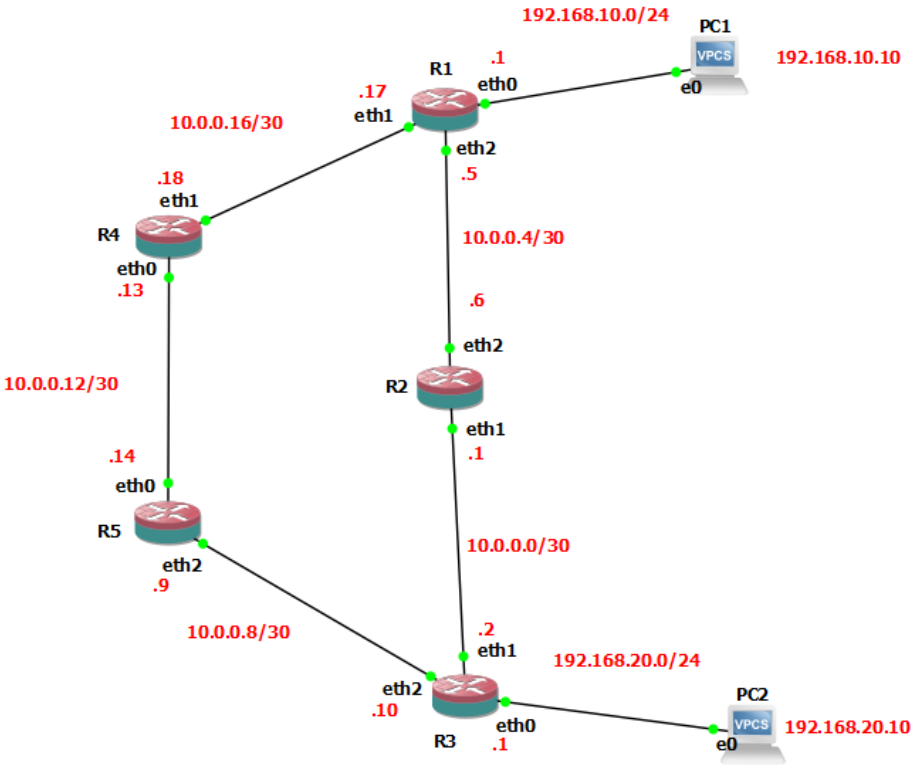
8.2.1 Srovnání výběru nejkratší cesty

Zapneme si program GNS3 a vytvoříme si nový projekt. Na obrázku 8-1 je zobrazená legenda levého menu.



Obr. 8-1: Legenda menu

V menu klikneme na ikonu směrovače, vybereme směrovač VyOS a vložíme ho na plochu pětkrát. Poté si klikneme na koncová zařízení, zařízení VPCS vložíme na plochu dvakrát a zapojíme jej podle obrázku 8-2.



Obr. 8-2: Topologie zapojení

Směrovače je možné připojit k libovolným portům. Je doporučeno je zapojit podle obrázku 8-1, v jiném případě budete muset využít jinou konfiguraci, než je zde v návodu. Když máme zařízení propojené, můžeme je zapnout pomocí tlačítka ve tvaru zeleného trojúhelníku v horní liště. Dvojklikem na každý směrovač si otevřeme konzole směrovačů R1 až R5. Stejně si otevřeme i konzole virtuálních počítačů PC1 a PC2. Přihlašovací jméno do směrovačů je *student* a heslo je *student*. Ve směrovačích a ve virtuálních počítačích teď není žádná konfigurace, proto ji tam musíme vložit. Začneme s konfigurací počítačů, protože ta je jednoduchá. Je potřebné pouze nastavit IP adresy, masky sítě a výchozí bránu. Pro toto nastavení do počítačů vložíme následující příkazy:

PC1> ip 192.168.10.10 255.255.255.0 192.168.10.1
PC2> ip 192.168.20.10 255.255.255.0 192.168.20.1

Pomocí příkazu *show ip* můžeme ověřit, že IP adresy jsou nastavené správně.

Na směrovačích si nejdřív nastavíme konfiguraci pro RIP protokol. Začneme se směrovačem R1. Po přihlášení vstoupíme do konfiguračního módu pomocí příkazu *configure* a potom nastavíme IP adresy na všechny jeho rozhrání a vypneme rozhraní, které nepoužíváme pomocí následujících příkazů:

set interfaces ethernet eth0 address 192.168.10.1/24
set interfaces ethernet eth1 address 10.0.0.17/30
set interfaces ethernet eth2 address 10.0.0.5/30
set interfaces ethernet eth3 disable
commit
save
exit

Změna v konfiguraci směrovače nastane až po použití příkazu *commit*. Příkaz *save* slouží na uložení konkrétní konfigurace do paměti a tato konfigurace bude načítaná v případě restartování zařízení. Pro výstup z konfiguračního módu a vstup do operačního módu se používá příkaz *exit*. V operačním módu si můžeme zkontrolovat, že jsou IP adresy nastavené správně, pomocí příkazu *show interfaces*. Na směrovači R1 nastavíme konfiguraci pro protokol RIP. Opět vstoupíme do konfiguračního módu pomocí příkazu *configure* a pro nakonfigurovaní RIP protokolu stačí nastavit IP adresy síti pro protokol RIP. Konfigurace pro směrovač R1 je následující:

set protocols rip network 192.168.10.0/24
set protocols rip network 10.0.0.0/8
commit
save
exit

Těmito příkazy jsme nastavili na směrovači R1 konfiguraci pro protokol RIP. Teď už je potřebné jen nastavit konfiguraci na směrovače R2 až R5. Konfigurace je stejná jako pro směrovač R1, jen je potřené změnit IP adresy na rozhraních a IP adresy sítí v konfiguraci RIP protokolu. Předtím, než nastavíte konfiguraci na směrovači R2, zapněte zachytávání paketů mezi směrovačem R1 a R2, abychom viděli, jak se navazuje sousedství a jaké informace si mezi sebou směrovače posílají. Zachytávání paketů zapneme tak, že pravým tlačítkem klikneme na linku a vybereme *Start capture* a poté klikneme na OK, tím se spustí program Wireshark.

Po nakonfigurovaní všech směrovačů by měly být virtuální počítače PC1 a PC2 schopné komunikovat. To můžeme ověřit pomocí pingu například z PC1:

PC1> ping 192.168.20.10

Pokud nekomunikují, snažte se zjistit, kde je chyba. Ve Wiresharku máme odchycené pakety toho, jak se navazovalo sousedství mezi směrovači R1 a R2, i toho, jak komunikují dále. Na obrázku 8-3 můžeme vidět odchycené pakety.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.0.0.5	224.0.0.9	RIPv2	66	Request
2	0.009655	10.0.0.5	224.0.0.22	IGMPv3	60	Membership Report / Join group 224.0.0.9 for any sources
3	0.635422	10.0.0.5	224.0.0.22	IGMPv3	60	Membership Report / Join group 224.0.0.9 for any sources
4	0.843639	10.0.0.5	224.0.0.9	RIPv2	86	Response
5	5.799183	10.0.0.6	224.0.0.9	RIPv2	66	Request
6	5.800064	10.0.0.5	10.0.0.6	RIPv2	86	Response
7	5.808795	10.0.0.6	224.0.0.22	IGMPv3	60	Membership Report / Join group 224.0.0.9 for any sources
8	6.530689	10.0.0.6	224.0.0.22	IGMPv3	60	Membership Report / Join group 224.0.0.9 for any sources
9	6.715649	10.0.0.6	224.0.0.9	RIPv2	66	Response
10	6.717279	10.0.0.5	224.0.0.9	RIPv2	66	Response
11	10.995464	0c:a4:7d:d6:d3:02	0c:a4:7d:32:09:02	ARP	60	Who has 10.0.0.6? Tell 10.0.0.5
12	10.996032	0c:a4:7d:32:09:02	0c:a4:7d:d6:d3:02	ARP	60	10.0.0.6 is at 0c:a4:7d:32:09:02
13	11.030089	10.0.0.6	224.0.0.9	RIPv2	86	Response
14	16.043100	10.0.0.5	224.0.0.9	RIPv2	66	Response
15	28.909708	10.0.0.5	224.0.0.9	RIPv2	106	Response
16	36.699505	10.0.0.6	224.0.0.9	RIPv2	106	Response

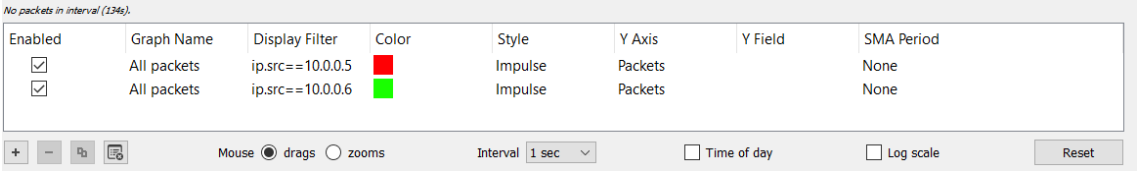
Obr. 8-3: Zachycené pakety při spuštění RIP protokolu

Ze zachyceného provozu vidíme, že první věcí, kterou směrovač při spuštění protokolu udělal, bylo odeslání *Request* paketu na multicast IP adresu 244.0.0.9. Když klikneme na paket, vidíme, že paket neobsahuje žádné informace o sítích. Obsahuje pouze informaci, že je to Request a že je to RIP verze 2. Podrobněji to můžeme vidět na obrázku 8-4.

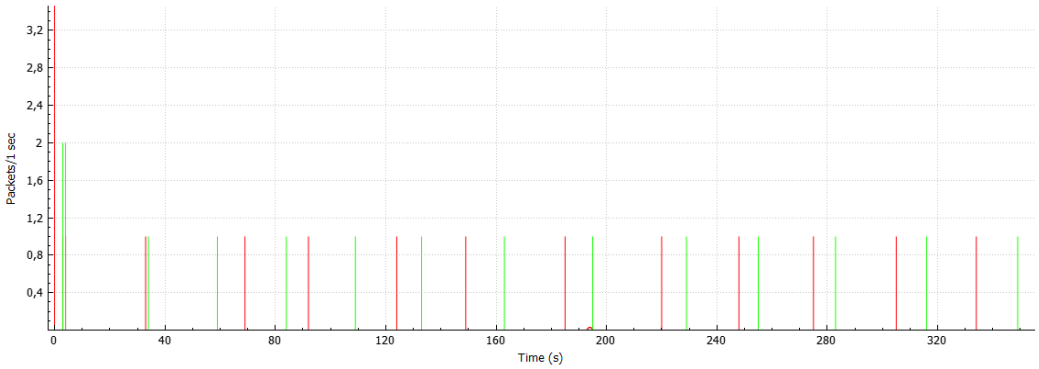
```
> Frame 1: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0
> Ethernet II, Src: 0c:a4:7d:d6:d3:02 (0c:a4:7d:d6:d3:02), Dst: IPv4mcast_09 (01:00:5e:00:00:09)
> Internet Protocol Version 4, Src: 10.0.0.5, Dst: 224.0.0.9
> User Datagram Protocol, Src Port: 520, Dst Port: 520
v Routing Information Protocol
    Command: Request (1)
    Version: RIPv2 (2)
  v Address not specified, Metric: 16
      Address Family: Unspecified (0)
      Route Tag: 0
      Netmask: 0.0.0.0
      Next Hop: 0.0.0.0
      Metric: 16
```

Obr. 8-4: Detail paketu Request

Když si zobrazíme odchycené pakety v grafu, můžeme vidět, že se na začátku poslalo nejvíc paketů a další pakety se posílaly každých 30 vteřin. Graf se zobrazí tím, že ve Wireshark klikneme na Statistics, vybereme I/O Graph a filtr nastavíme jako na obrázku 8-5. Na obrázku 8-6 je samotný graf.



Obr. 8-5: Nastavení filtru



Obr. 8-7: Grafické zobrazení paketů RIP

Dále se směrovač připájí do multicast skupiny 224.0.0.9, protože RIP verze 2 funguje na multicastu na rozdíl od RIP verze 1, která funguje na broadcastu. Poté opět směrovač R1 posílá Response paket, který se posílá každých 30 sekund. Na detail Response paketu se můžeme podívat na obrázku 8-7.

```
> Frame 4: 86 bytes on wire (688 bits), 86 bytes captured (688 bits) on interface 0
> Ethernet II, Src: 0c:a4:7d:d6:d3:02 (0c:a4:7d:d6:d3:02), Dst: IPv4mcast_09 (01:00:5e:00:00:09)
> Internet Protocol Version 4, Src: 10.0.0.5, Dst: 224.0.0.9
> User Datagram Protocol, Src Port: 520, Dst Port: 520
v Routing Information Protocol
    Command: Response (2)
    Version: RIPv2 (2)
  v IP Address: 10.0.0.16, Metric: 1
      Address Family: IP (2)
      Route Tag: 0
      IP Address: 10.0.0.16
      Netmask: 255.255.255.252
      Next Hop: 0.0.0.0
      Metric: 1
  > IP Address: 192.168.10.0, Metric: 1
```

Obr. 8-6: Detail paketu Response

V paketu Response se posílá informace o tom, že to je paket typu Response, a stejně jako v případě Request se zde posílá informace o tom, že je to RIP verze 2. Rozdíl je v tom, že v paketu Response se už posílají informace o sítích. Konkrétně v tomto případě jsou to jen sítě přímo připojené do směrovače, to víme podle toho, že je metrika 1. V informaci o síti se kromě samotné IP adresy sítě přenáší také maska sítě, next hop a metrika. Dále můžeme vidět. jak směrovač R2 posílá Request paket, který je stejný jako v případě směrovače R1 a na něj mu směrovač R1 odpovídá paketem Response. Obsah informace v paketu Response je stejný jako na obrázku 8-7, rozdíl je jen v tom, že tento paket je přímo poslaný na IP adresu rozhraní eth2 směrovače R2 a není poslaný multicastem.

Podle teorie by se v případě protokolu RIP měla nejkratší cesta vybírat podle počtu skoků, takže si to vyzkoušíme. Otevřeme si konzoli počítače PC1 a zadáme do něho příkaz *trace*, který nám ukáže, jak se dostat k počítači PC2:

PC1> trace 192.168.20.10

Na obrázku 8-8 vidíme výstup z příkazu trace.

```
PC1> trace 192.168.20.10
trace to 192.168.20.10, 8 hops max, press Ctrl+C to stop
 1   192.168.10.1   0.938 ms  0.704 ms  1.118 ms
 2   10.0.0.6   2.050 ms  2.138 ms  2.152 ms
 3   10.0.0.2   3.749 ms  3.372 ms  4.001 ms
 4   *192.168.20.10   6.672 ms (ICMP type:3, code:3, Destination port unreachable)
```

Obr. 8-8: Hledání cesty od PC1 k PC2

Vidíme, že první směrovač má IP adresu 192.168.10.1, to je směrovač R1, dále následuje IP adresa 10.0.0.6, to je směrovač R2. Posléze je tam IP adresa 10.0.0.2, která patří směrovači R3, nakonec vidíme IP adresu 192.168.20.10, která patří našemu cíli PC2. Takže můžeme vidět, že si směrovač pomocí protokolu skutečně zvolil tu nejlepší cestu a to tu nejkratší přes směrovače R1, R2 a R3. Dále si můžeme zobrazit směrovací tabulku směrovačů. Jako příklad si zobrazíme směrovací tabulku směrovače R1. Na to použijeme příkaz v konzoli směrovače R1:

show ip route

Ten nám zobrazí všechny směrovací informace, které směrovač má. Konkrétní výstup můžeme vidět na obrázku 8-9.

```
student@student:~$ show ip route
Codes: K - kernel route, C - connected, S - static, R - RIP,
       O - OSPF, I - IS-IS, B - BGP, E - EIGRP, N - NHRP,
       T - Table, v - VNC, V - VNC-Direct, A - Babel, D - SHARP,
       F - PBR, f - OpenFabric,
       > - selected route, * - FIB route, q - queued route, r - rejected route

R>* 10.0.0.0/30 [120/2] via 10.0.0.6, eth2, 00:46:38
C>* 10.0.0.4/30 is directly connected, eth2, 00:51:15
R>* 10.0.0.8/30 [120/3] via 10.0.0.6, eth2, 00:46:34
R>* 10.0.0.12/30 [120/2] via 10.0.0.18, eth1, 00:46:30
C>* 10.0.0.16/30 is directly connected, eth1, 00:51:16
C>* 192.168.10.0/24 is directly connected, eth0, 00:51:14
R>* 192.168.20.0/24 [120/3] via 10.0.0.6, eth2, 00:46:34
```

Obr. 8-9: Směrovací tabulka směrovače R1

V této tabulce vidíme sítě přímo připojené, ty jsou označené jako C (connected), a vidíme taky sítě naučené pomocí protokolu RIP, ty jsou označené jako R (RIP). Kdybychom si chtěli zobrazit směrovací tabulku pouze se sítěmi naučenými pomocí protokolu RIP, použijeme příkaz *show ip route rip*. V hranaté závorce za adresou sítě můžeme vidět, jaká je cena cesty do dané sítě. Pro příklad si vezmeme první řádek:

R>* 10.0.0.0/30 [120/2] via 10.0.0.6, eth2, 00:51:35

Podle toho, co jsme si již řekli, víme, že první písmeno výše uvedeného řádku znamená, že se směrovač naučil o dostupnosti sítě pomocí protokolu RIP. Dále následuje adresa sítě a prefix její masky. V hranaté závorce [120/2] určuje první číslici priorita směrovacího protokolu RIP. Kdyby bylo na směrovači více směrovacích protokolu, vybral by protokol s nejnižším číslem priority. Druhá číslice určuje samotnou cenu cesty naučenou pomocí protokolu RIP, konkrétně u tohoto protokolu je to počet skoků.

Pomocí protokolu RIP si můžeme na směrovači zobrazit sítě naučené tímto protokolem. Na to použijeme příkaz:

show ip rip

Výstup příkazu pro směrovač R1 můžeme vidět na obrázku 8-10.

```
student@student:~$ show ip rip
Codes: R - RIP, C - connected, S - Static, O - OSPF, B - BGP
Sub-codes:
      (n) - normal, (s) - static, (d) - default, (r) - redistribute,
      (i) - interface

     Network            Next Hop         Metric From            Tag Time
R(n) 10.0.0.0/30        10.0.0.6              2 10.0.0.6         0 02:55
C(i) 10.0.0.4/30        0.0.0.0               1 self             0
R(n) 10.0.0.8/30        10.0.0.6              3 10.0.0.6         0 02:55
R(n) 10.0.0.12/30       10.0.0.18             2 10.0.0.18        0 02:47
C(i) 10.0.0.16/30       0.0.0.0               1 self             0
C(i) 192.168.10.0/24    0.0.0.0               1 self             0
R(n) 192.168.20.0/24    10.0.0.6              3 10.0.0.6         0 02:55
```

Obr. 8-10: Směrovací informace protokolu RIP

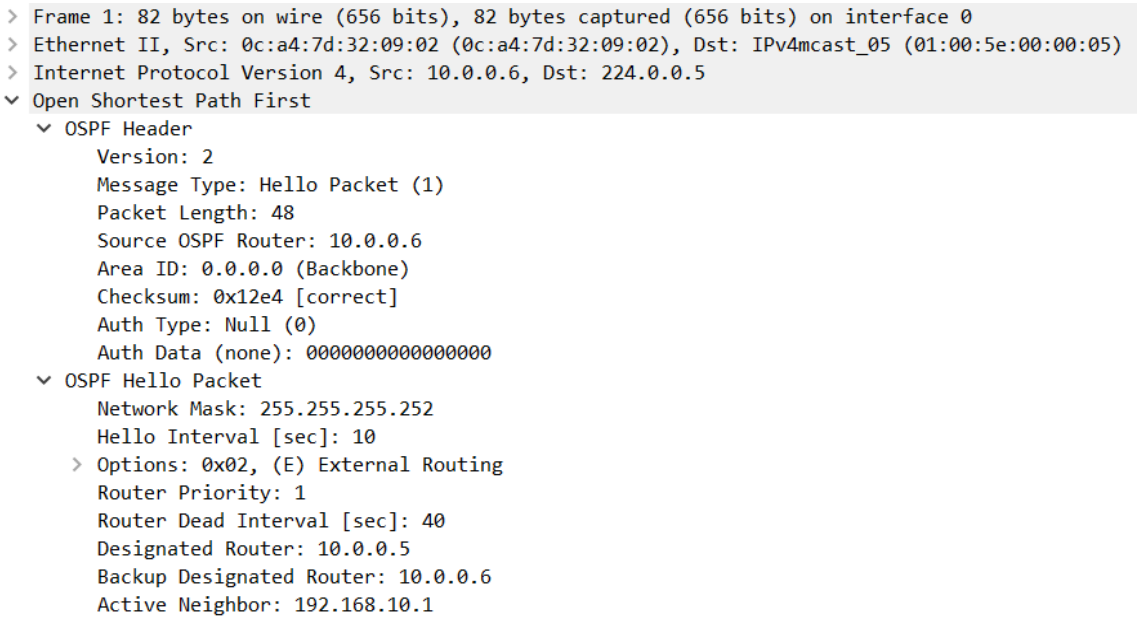
Na tomto výstupu můžeme vidět všechny sítě naučené pomocí protokolu RIP, next hop směrovač, metriku, od koho se směrovač R1 tuto síť naučil a čas, kdy vyprší platnost tohoto záznamu.

Nyní si na stejné směrovače nakonfigurujeme směrovací protokol OSPF. Protokol RIP nemusíme z konfigurace vymazávat, protože OSPF má ve směrovačích přednost před směrovacím protokolem RIP. Konfigurace protokolu OSPF je téměř stejná jako konfigurace protokolu RIP. Jediný rozdíl je v tom, že v případě OSPF je potřebné nadefinovat, do které oblasti chceme danou síť přiřadit. V našem případě budeme používat jen jednu oblast, a to oblast číslo 0, takzvanou backbone oblast. Víc oblastí v OSPF budeme používat v další části laboratorní úlohy. Pro konfiguraci nejdřív vejdeme do konfiguračního módu a použijeme následující příkazy, abychom konfigurovali připojené sítě. Zde je uvedený příklad pro směrovač R1:

configure
set protocols ospf area 0 network 192.168.10.0/24
set protocols ospf area 0 network 10.0.0.0/8
commit
save
exit

Předtím, než začneme zadávat příkazy pro OSPF do dalších směrovačů, si zapneme zachytávání paketů mezi směrovačem R1 a R2. Stejně jako v případě RIP nemusíme zadávat všechny IP adresy sítí přesně. Například jsou do směrovače připojené dvě sítě - 10.0.0.16/30 a 10.0.0.4/30, v tomto případě stačí zadat síť 10.0.0.0/8, protože obě sítě spadají pod tento adresný rozsah. Taky je možné zadat jen názvy rozhraní, kde chceme mít zapnuté OSPF směrování, a směrovač si adresy sítí zjistí z rozhraní. Stejným způsobem nastavíme OSPF pro ostatní směrovače.

Teď, když máme nakonfigurované všechny směrovače, se zkusíme podívat, jak mezi sebou směrovače předávají informace o sítích a aktualizace, když nějaká síť vypadne, nebo naopak když se nová síť přidá do topologie. Ve Wiresharku máme odchycené pakety komunikace mezi směrovačem R1 a R2. Můžeme si nastavit filtr pouze na protokol OSPF. Vybereme si jeden z Hello paketů a klikneme na něj. Příklad je na obrázku 8-11.



Obr. 8-11: Hello paket

Jak můžeme na první pohled vidět, rozdíl mezi RIP pakety a OSPF pakety je v tom, že RIP běží nad transportním protokolem UDP, zatímco protokol OSPF běží přímo nad protokolem IP. Pakety protokolu OSPF se vždy skládají z hlavičky a těla. V hlavičce se nachází verze OSPF protokolu, typ OSPF zprávy, délka paketu, zdrojová adresa směrovače (tzv. Router-ID), OSPF oblast, kontrolní součet a autentifikace. Existuje více typů OSPF zpráv. Hello paket slouží k tomu, aby se udržovalo sousedství mezi směrovači. Když směrovač od svého souseda nedostane za určitou dobu Hello paket, pokládá souseda za nedostupného. V těle Hello paketu se nachází maska sítě, interval posílání Hello paketů, v možnostech se posílají schopnosti, které daný směrovač dokáže, dead interval a dále obsahuje informace o směrovačích v dané síti. Na to, aby se vytvořilo sousedství mezi dvěma směrovači, musí mít směrovače nastavený stejný interval zasílání Hello paketů, dead interval a musí se shodovat autentifikační informace.

Dále se podíváme, jak se vytváří sousedství. Na obrázku 8-12 vidíme odchycené pakety ve Wiresharku.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.0.0.6	224.0.0.5	OSPF	82	Hello Packet
2	9.171016	10.0.0.5	224.0.0.5	OSPF	78	Hello Packet
6	10.001260	10.0.0.6	224.0.0.5	OSPF	82	Hello Packet
9	10.013213	10.0.0.5	10.0.0.6	OSPF	66	DB Description
13	10.023371	10.0.0.6	10.0.0.5	OSPF	66	DB Description
14	10.023856	10.0.0.6	10.0.0.5	OSPF	246	DB Description
17	10.028019	10.0.0.5	10.0.0.6	OSPF	66	DB Description
18	10.028408	10.0.0.5	10.0.0.6	OSPF	82	LS Request
19	10.029695	10.0.0.6	10.0.0.5	OSPF	66	DB Description
20	10.030302	10.0.0.6	224.0.0.5	OSPF	238	LS Update
26	11.008888	10.0.0.5	224.0.0.5	OSPF	78	LS Acknowledge
31	14.715157	10.0.0.5	10.0.0.6	OSPF	70	LS Request
32	14.716227	10.0.0.5	224.0.0.5	OSPF	182	LS Update
33	14.717443	10.0.0.6	224.0.0.5	OSPF	110	LS Update
34	14.719695	10.0.0.5	224.0.0.5	OSPF	122	LS Update
35	15.009552	10.0.0.5	224.0.0.5	OSPF	78	LS Acknowledge
36	15.023634	10.0.0.6	10.0.0.5	OSPF	126	LS Update
37	15.038550	10.0.0.6	224.0.0.5	OSPF	78	LS Acknowledge
38	16.009817	10.0.0.5	224.0.0.5	OSPF	98	LS Acknowledge
39	19.171342	10.0.0.5	224.0.0.5	OSPF	82	Hello Packet
40	19.692960	10.0.0.6	224.0.0.5	OSPF	110	LS Update
41	20.001664	10.0.0.6	224.0.0.5	OSPF	82	Hello Packet
42	20.010781	10.0.0.5	224.0.0.5	OSPF	78	LS Acknowledge
43	20.029427	10.0.0.5	10.0.0.6	OSPF	122	LS Update
44	20.043020	10.0.0.6	224.0.0.5	OSPF	78	LS Acknowledge
46	29.171863	10.0.0.5	224.0.0.5	OSPF	82	Hello Packet
47	30.001009	10.0.0.6	224.0.0.5	OSPF	82	Hello Packet
48	39.172133	10.0.0.5	224.0.0.5	OSPF	82	Hello Packet
49	40.001834	10.0.0.6	224.0.0.5	OSPF	82	Hello Packet
50	49.090652	10.0.0.5	224.0.0.5	OSPF	122	LS Update
51	49.172077	10.0.0.5	224.0.0.5	OSPF	82	Hello Packet
52	50.005088	10.0.0.6	224.0.0.5	OSPF	82	Hello Packet
53	50.057136	10.0.0.6	224.0.0.5	OSPF	78	LS Acknowledge
55	59.173014	10.0.0.5	224.0.0.5	OSPF	82	Hello Packet
56	60.005386	10.0.0.6	224.0.0.5	OSPF	82	Hello Packet

Obr. 8-12: Vytváření sousedství v OSPF

Jak můžeme vidět, nejdříve se posílají Hello pakety a poté si směrovače pomocí zprávy type DB Description navzájem porovnávají své OSPF databáze. Nakonec si posílají pakety typu LS Request a LS Update a potvrzují si přijetí paketů pomocí zprávy type LS Acknowledge.

V případě že je v jedné OSPF síti připojených více směrovačů (přes přepínač), probíhají mezi nimi volby pro výběr takzvaného designated směrovače a jeho záloha. Slouží to k tomu, aby se v síti posílaly směrovací informace jen mezi designated směrovačem a ostatními směrovači. Tím se v OSPF síti výrazně sníží provoz. V případě že jsou směrovače přímo propojené, je dobré nastavit síť jako point-to-point, protože v síti jsou jen dva směrovače, a tudíž není potřeba, aby probíhaly volby výběru designated směrovače. Na VyOS se OSPF síť typu point-to-point nastavuje na rozhraní. Příklad pro směrovač R1:

configure
set interfaces ethernet eth1 ip ospf network point-to-point
set interfaces ethernet eth2 ip ospf network point-to-point
commit
save
exit

Taky je dobré v případě OSPF nastavit některé rozhraní jako pasivní, to znamená, že se do sítě nebudou posílat Hello pakety. Nastavuje se to například do sítě, ve které se nacházejí koncová zařízení. Příklad nastavení na směrovači R1:

configure
set protocols ospf passive-interface eth0
commit
save
exit

U protokolu RIP se funkce volby designated směrovače nenachází.

8.2.2 Změna cesty OSPF pomocí šířky pásma

Protokol OSPF vybírá nejkratší cestu výpočtem z parametru bandwidth, který je nastaven v odchozím rozhraní. Parametr bandwidth nemusí mít nastavenou reálnou šířku pásma, která je na rozhraní, a protokol OSPF ani automaticky nenastavuje parametr bandwidth podle reálné rychlosti na rozhraní. Pokud bandwidth na rozhraní změníme, můžeme tím dosáhnout toho, že se změní cesta, kterou budou směrovače posílat pakety. Ukážeme si to na příkladu z topologie 8-2, kdy chceme, aby směrovače posílaly pakety přes směrovače R4 a R5 místo toho, aby je posílaly přes směrovač R2, tak, že změníme bandwidth na rozhraní směrovače R3 směrem ke směrovači R2.

Jako první si otevřeme konzoli směrovače R3 a zkontrolujeme, bandwidth je nastaven na rozhraní eth1. To zjistíme pomocí příkazu:

show ip ospf interface eth1

Na obrázku 8-13 je zobrazen výstup předchozího příkazu.

```
student@student:~$ show ip ospf interface eth1
eth1 is up
  ifindex 3, MTU 1500 bytes, BW 1000 Mbit <UP,BROADCAST,RUNNING,MULTICAST>
  Internet Address 10.0.0.2/30, Broadcast 10.0.0.3, Area 0.0.0.0
  MTU mismatch detection: enabled
  Router ID 192.168.20.1, Network Type BROADCAST, Cost: 100
  Transmit Delay is 1 sec, State DR, Priority 1
  Backup Designated Router (ID) 10.0.0.6, Interface Address 10.0.0.1
  Multicast group memberships: OSPFAllRouters OSPFDesignatedRouters
  Timer intervals configured, Hello 10s, Dead 40s, Wait 40s, Retransmit 5
    Hello due in 3.475s
  Neighbor Count is 1, Adjacent neighbor count is 1
```

Obr. 8-13: Informace o OSPF rozhraní

Jak můžeme na obrázku vidět, tak rozhraní bandwidth je nastavené na 1000 Mbit. Zkusíme hodnotu bandwidth změnit na takovou hodnotu, aby směrovače posílaly pakety přes směrovače R4 a R5. Následující příkaz změní bandwidth rozhraní eth1 na hodnotu 400 Mbit (musí být zadán v konfiguračním režimu):

set interfaces ethernet eth1 ip ospf bandwidth 400

Změnu rychlosti bandwidth je potřebné udělat jak na směrovači R3, tak na směrovači R2. Konfiguraci potvrdíme a uložíme a vrátíme se zpátky do operačního kódu. Tam si opět zobrazíme informaci o rozhraní eth1 a zkontrolujeme, jestli se bandwidth změnilo. Na obrázku 8-14 můžeme vidět, že je nyní nastaven na hodnotu 400 Mbit.

```
student@student:~$ show ip ospf interface eth1
eth1 is up
  ifindex 3, MTU 1500 bytes, BW 400 Mbit <UP,BROADCAST,RUNNING,MULTICAST>
  Internet Address 10.0.0.2/30, Broadcast 10.0.0.3, Area 0.0.0.0
  MTU mismatch detection: enabled
  Router ID 192.168.20.1, Network Type BROADCAST, Cost: 250
  Transmit Delay is 1 sec, State DR, Priority 1
  Backup Designated Router (ID) 10.0.0.6, Interface Address 10.0.0.1
  Multicast group memberships: OSPFAllRouters OSPFDesignatedRouters
  Timer intervals configured, Hello 10s, Dead 40s, Wait 40s, Retransmit 5
    Hello due in 0.327s
  Neighbor Count is 1, Adjacent neighbor count is 1
```

Obr. 8-14: Informace o rozhraní eth1

Vidíme, že se bandwidth změnilo na hodnotu 400 Mbit. Dále si zkontrolujeme, jestli se změnila směrovací tabulka směrovače R3. Na obrázku 8-15 je směrovací tabulka směrovače R3.

```
student@student:~$ show ip route ospf
Codes: K - kernel route, C - connected, S - static, R - RIP,
       O - OSPF, I - IS-IS, B - BGP, E - EIGRP, N - NHRP,
       T - Table, v - VNC, V - VNC-Direct, A - Babel, D - SHARP,
       F - PBR, f - OpenFabric,
       > - selected route, * - FIB route, q - queued route, r - rejected route

O   10.0.0.0/30 [110/250] is directly connected, eth1, 00:09:54
O>* 10.0.0.4/30 [110/350] via 10.0.0.1, eth1, 00:09:54
O   10.0.0.8/30 [110/100] is directly connected, eth2, 00:28:01
O>* 10.0.0.12/30 [110/200] via 10.0.0.9, eth2, 00:27:10
O>* 10.0.0.16/30 [110/300] via 10.0.0.9, eth2, 00:13:11
O>* 192.168.10.0/24 [110/400] via 10.0.0.9, eth2, 00:09:54
O   192.168.20.0/24 [110/100] is directly connected, eth0, 00:28:01
```

Obr. 8-15: Směrovací tabulka R3

Jak můžeme na obrázku 8-15 vidět, směrovač si do sítě 192.168.10.10/24 vybírá cestu přes rozhraní eth2, což znamená, že teď se budou pakety posílat přes směrovače R4 a R5 místo směrovače R2. Můžeme si to vyzkoušet na virtuálním počítači PC1 nebo PC2. Na obrázku 8-16 je ukázka trasování cesty z počítače PC2 na počítač PC1.

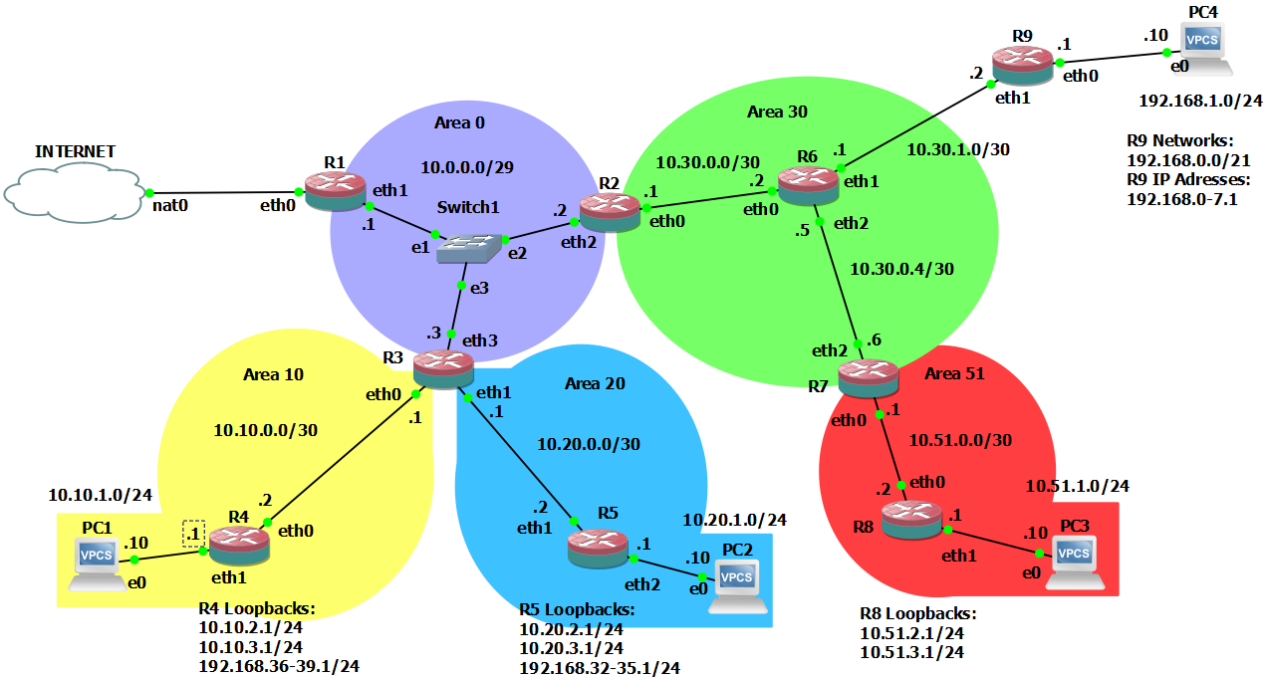
```
PC2> trace 192.168.10.10
trace to 192.168.10.10, 8 hops max, press Ctrl+C to stop
 1   192.168.20.1   2.171 ms  1.733 ms  1.570 ms
 2   10.0.0.9   2.986 ms  2.344 ms  2.953 ms
 3   10.0.0.13   2.698 ms  2.521 ms  1.845 ms
 4   10.0.0.17   3.253 ms  2.649 ms  2.117 ms
 5   *192.168.10.10   3.839 ms (ICMP type:3, code:3, Destination port unreachable)
```

Obr. 8-16: Trasování z PC2 na PC1

8.3 OSPF – funkcionality

V této části úlohy si vyzkoušíme v simulátoru GNS3 některé funkcionality, které protokol OSPF nabízí. Mezi ně patří například oblasti, sumarizace sítí v oblasti, stub oblasti, virtuální link, redistribuce sítí do OSPF a další.

V menu GNS3 vybereme File -> New blank project, dále klikneme na záložku Project library, klikneme na projekt s názvem OSPF-F a vytvoříme jeho kopii pomocí tlačítka Duplicate. Topologie projektu by měla vypadat jako na obrázku 8-17.



Obr. 8-17: Topologie sítě

Síť je rozdělena na pět oblastí - 0, 10, 20, 30 a 51. Na směrovačích jsou nakonfigurované IP adresy a sítě jsou přidané do správných OSPF oblastí. Všechny směrovače kromě R8 a R9 by měly být schopné navzájem komunikovat. Směrovač R8 je v oblasti 51, která není přímo spojená s oblastí 0, a směrovač R9 nemá zapnutý žádný OSPF proces, funguje jen na statickém směrování. Do všech směrovačů se můžeme přihlásit pod uživatelským jménem *student* a heslem *student*. Je doporučeno si otevřít konzole všech směrovačů v síti.

8.3.1 Sumarizace sítí

Jako první si ukážeme, jak funguje sumarizace sítě u protokolu OSPF. Sumarizace sítě je možné nastavit jen na hraničních směrovačích, protože směrovače, které se nachází uvnitř sítě, potřebují celou směrovací tabulku oblasti, ve které se nacházejí. V topologii je možné sumarizovat oblasti 0, 10, 20, 30. Všechny oblasti mají IP adresy přidělené podle následujícího vzorce: 10.[číslo oblasti].0.0/16. V oblasti 10 se nachází i sítě s IP adresami 192.168.36-39.0/24 a v oblasti 20 se nachází sítě s IP adresami 192.168.32-35.0/24.

Nejdřív si otevřeme konzoli směrovače R1 a zobrazíme si jeho směrovací tabulku pomocí následujícího příkazu:

show ip route

Výstup by měl vypadat jako na obrázku 8-18.

```
student@R1:~$ show ip route
Codes: K - kernel route, C - connected, S - static, R - RIP,
       O - OSPF, I - IS-IS, B - BGP, E - EIGRP, N - NHRP,
       T - Table, v - VNC, V - VNC-Direct, A - Babel, D - SHARP,
       F - PBR, f - OpenFabric,
       > - selected route, * - FIB route, q - queued route, r - rejected route

S>* 0.0.0.0/0 [210/0] via 192.168.122.1, eth0, 00:31:28
O   10.0.0.0/29 [110/100] is directly connected, eth1, 00:31:16
C>* 10.0.0.0/29 is directly connected, eth1, 00:31:44
O>* 10.10.0.0/16 [110/300] via 10.0.0.3, eth1, 00:29:54
O>* 10.20.0.0/30 [110/200] via 10.0.0.3, eth1, 00:29:57
O>* 10.20.1.0/24 [110/300] via 10.0.0.3, eth1, 00:29:57
O>* 10.20.2.0/24 [110/210] via 10.0.0.3, eth1, 00:29:57
O>* 10.20.3.0/24 [110/210] via 10.0.0.3, eth1, 00:29:57
O>* 10.30.0.0/30 [110/200] via 10.0.0.2, eth1, 00:30:12
O>* 10.30.0.4/30 [110/300] via 10.0.0.2, eth1, 00:30:12
O>* 10.30.1.0/30 [110/300] via 10.0.0.2, eth1, 00:30:12
O>* 192.168.32.0/24 [110/210] via 10.0.0.3, eth1, 00:29:57
O>* 192.168.33.0/24 [110/210] via 10.0.0.3, eth1, 00:29:57
O>* 192.168.34.0/24 [110/210] via 10.0.0.3, eth1, 00:29:57
O>* 192.168.35.0/24 [110/210] via 10.0.0.3, eth1, 00:29:57
O>* 192.168.36.0/24 [110/210] via 10.0.0.3, eth1, 00:29:54
O>* 192.168.37.0/24 [110/210] via 10.0.0.3, eth1, 00:29:54
O>* 192.168.38.0/24 [110/210] via 10.0.0.3, eth1, 00:29:54
O>* 192.168.39.0/24 [110/210] via 10.0.0.3, eth1, 00:29:54
C>* 192.168.122.0/24 is directly connected, eth0, 00:31:30
```

Obr. 8-18: Směrovací tabulka směrovače R1

Z obrázku vidíme, že síť 10.10.0.0/16 je už sumarizovaná, takže už potřebujeme sumarizovat jen sítě 10.20.0.0/16, 10.30.0.0/16 a 192.168.32-39.0/24. Oblast 20 je připojená do směrovače R3, proto musíme sumarizaci nakonfigurovat na něm. Nakonfigurujeme ji tak, že vstoupíme do konfiguračního módu a zadáme následující příkazy:

configure
set protocols ospf area 0 range 10.0.0.0/16
set protocols ospf area 20 range 10.20.0.0/16
set protocols ospf area 10 range 192.168.36.0/22
set protocols ospf area 20 range 192.168.32.0/22
commit
save
exit

Těmito příkazy jsme sumarizovali sítě v oblastech 10 a 20 a oblast 0. Sítě z oblastí 10 a 20 uvidíme v ostatních sítích jako sumarizované a v oblastech 10 a 20 uvidíme síť oblasti 0 jako 10.0.0.0/16 namísto sítě 10.0.0.0/29. Když si teď zobrazíme směrovací tabulku směrovače R1, měla vypadat jako na obrázku 8-19.

```
student@R1:~$ show ip route
Codes: K - kernel route, C - connected, S - static, R - RIP,
       O - OSPF, I - IS-IS, B - BGP, E - EIGRP, N - NHRP,
       T - Table, v - VNC, V - VNC-Direct, A - Babel, D - SHARP,
       F - PBR, f - OpenFabric,
       > - selected route, * - FIB route, q - queued route, r - rejected route

S>* 0.0.0.0/0 [210/0] via 192.168.122.1, eth0, 00:45:13
O   10.0.0.0/29 [110/100] is directly connected, eth1, 00:45:01
C>* 10.0.0.0/29 is directly connected, eth1, 00:45:29
O>* 10.10.0.0/16 [110/300] via 10.0.0.3, eth1, 00:43:39
O>* 10.20.0.0/16 [110/300] via 10.0.0.3, eth1, 00:03:39
O>* 10.30.0.0/30 [110/200] via 10.0.0.2, eth1, 00:43:57
O>* 10.30.0.4/30 [110/300] via 10.0.0.2, eth1, 00:43:57
O>* 10.30.1.0/30 [110/300] via 10.0.0.2, eth1, 00:43:57
O>* 192.168.32.0/22 [110/210] via 10.0.0.3, eth1, 00:03:39
O>* 192.168.36.0/22 [110/210] via 10.0.0.3, eth1, 00:03:39
C>* 192.168.122.0/24 is directly connected, eth0, 00:45:15
```

Obr. 8-19: Směrovací tabulka směrovače R1 po sumarizaci

Nyní můžeme vidět, že sítě 10.10.0.0/16, 10.20.0.0/16, 192.168.32.0/22 a 192.168.36.0/22 jsou teď sumarizovány. Pořád nemáme sumarizovanou síť 10.30.0.0. Tuto síť je potřebné sumarizovat na směrovači R2. Příkazy jsou stejné jako v případě směrovače R3 jen s jinými IP adresami sítě a jinou OSPF oblastí.

8.3.2 Redistribuce sítí do OSPF

Do protokolu OSPF je možné redistribuovat sítě z jiného směrovacího protokolu nebo staticky nastavené sítě. Taky se využívá distribuce výchozí brány, a to v případě že ze směrovacího protokolu OSPF vede jen jedna cesta ven. V našem případě je výchozí brána směrovač R1, který je připojený k internetu. Do směrovacího protokolu OSPF chceme taky redistribuovat sítě ze směrovače R9, který nemá zapnutý směrovací protokol OSPF.

Otevřeme si konzoli některého směrovače kromě směrovačů R1, R8 a R9. Jako příklad si ukážeme směrovací tabulku směrovače R4 na obrázku 8-20.

```
student@R4:~$ show ip route
Codes: K - kernel route, C - connected, S - static, R - RIP,
       O - OSPF, I - IS-IS, B - BGP, E - EIGRP, N - NHRP,
       T - Table, v - VNC, V - VNC-Direct, A - Babel, D - SHARP,
       F - PBR, f - OpenFabric,
       > - selected route, * - FIB route, q - queued route, r - rejected route

O>* 10.0.0.0/16 [110/200] via 10.10.0.1, eth0, 00:17:36
O   10.10.0.0/30 [110/100] is directly connected, eth0, 00:58:26
C>* 10.10.0.0/30 is directly connected, eth0, 00:58:47
O   10.10.1.0/24 [110/100] is directly connected, eth1, 00:58:26
C>* 10.10.1.0/24 is directly connected, eth1, 00:58:43
O   10.10.2.0/24 [110/10] is directly connected, dum2, 00:58:26
C>* 10.10.2.0/24 is directly connected, dum2, 00:59:00
O   10.10.3.0/24 [110/10] is directly connected, dum3, 00:58:26
C>* 10.10.3.0/24 is directly connected, dum3, 00:58:55
O>* 10.20.0.0/16 [110/300] via 10.10.0.1, eth0, 00:17:36
O>* 10.30.0.0/30 [110/300] via 10.10.0.1, eth0, 00:57:36
O>* 10.30.0.4/30 [110/400] via 10.10.0.1, eth0, 00:57:36
O>* 10.30.1.0/30 [110/400] via 10.10.0.1, eth0, 00:57:36
O>* 192.168.32.0/22 [110/210] via 10.10.0.1, eth0, 00:17:36
O   192.168.36.0/24 [110/10] is directly connected, dum36, 00:58:26
C>* 192.168.36.0/24 is directly connected, dum36, 00:59:05
O   192.168.37.0/24 [110/10] is directly connected, dum37, 00:58:26
C>* 192.168.37.0/24 is directly connected, dum37, 00:58:50
O   192.168.38.0/24 [110/10] is directly connected, dum38, 00:58:26
C>* 192.168.38.0/24 is directly connected, dum38, 00:58:57
O   192.168.39.0/24 [110/10] is directly connected, dum39, 00:58:26
C>* 192.168.39.0/24 is directly connected, dum39, 00:59:03
```

Obr. 8-20: Směrovací tabulka směrovače R4

Jak můžeme vidět, směrovač nemá nastavenou žádnou výchozí bránu (nemá cestu do sítě 0.0.0.0/0).

Na směrovači R1 vstoupíme do konfiguračního módu a nakonfigurujeme distribuci výchozí brány pomocí následujícího příkazu:

configure
set protocols ospf default-information originate
commit
save
exit

Když se teď podíváme na směrovací tabulku směrovače R4, uvidíme, že už má nastavenou výchozí bránu. Směrovací tabulka je na obrázku 8-21.

```
student@R4:~$ show ip route
Codes: K - kernel route, C - connected, S - static, R - RIP,
       O - OSPF, I - IS-IS, B - BGP, E - EIGRP, N - NHRP,
       T - Table, v - VNC, V - VNC-Direct, A - Babel, D - SHARP,
       F - PBR, f - OpenFabric,
       > - selected route, * - FIB route, q - queued route, r - rejected route

O>* 0.0.0.0/0 [110/10] via 10.10.0.1, eth0, 00:06:05
O>* 10.0.0.0/16 [110/200] via 10.10.0.1, eth0, 00:27:32
O   10.10.0.0/30 [110/100] is directly connected, eth0, 01:08:22
C>* 10.10.0.0/30 is directly connected, eth0, 01:08:43
O   10.10.1.0/24 [110/100] is directly connected, eth1, 01:08:22
C>* 10.10.1.0/24 is directly connected, eth1, 01:08:39
O   10.10.2.0/24 [110/10] is directly connected, dum2, 01:08:22
C>* 10.10.2.0/24 is directly connected, dum2, 01:08:56
O   10.10.3.0/24 [110/10] is directly connected, dum3, 01:08:22
C>* 10.10.3.0/24 is directly connected, dum3, 01:08:51
O>* 10.20.0.0/16 [110/300] via 10.10.0.1, eth0, 00:27:32
O>* 10.30.0.0/16 [110/400] via 10.10.0.1, eth0, 00:00:04
O>* 192.168.32.0/22 [110/210] via 10.10.0.1, eth0, 00:27:32
O   192.168.36.0/24 [110/10] is directly connected, dum36, 01:08:22
C>* 192.168.36.0/24 is directly connected, dum36, 01:09:01
O   192.168.37.0/24 [110/10] is directly connected, dum37, 01:08:22
C>* 192.168.37.0/24 is directly connected, dum37, 01:08:46
O   192.168.38.0/24 [110/10] is directly connected, dum38, 01:08:22
C>* 192.168.38.0/24 is directly connected, dum38, 01:08:53
O   192.168.39.0/24 [110/10] is directly connected, dum39, 01:08:22
C>* 192.168.39.0/24 is directly connected, dum39, 01:08:59
```

Obr. 8-21: Směrovací tabulka směrovače R4 po distribuci výchozí brány

Teď by všechny směrovače kromě R8 a R9 (kromě rozhraní eth1) měly být schopné komunikovat s celým internetem. Můžeme to ověřit pomocí pingu na adresu 8.8.8.8.

Dále nakonfigurujeme směrovač R6, který má nastavenou statickou cestu k sítím směrovače R9, aby ji redistribuoval do celé OSPF sítě. Tím budou mít všechny směrovače v OSPF síti cestu ke směrovači R9 a budou moci komunikovat se všemi sítěmi, které jsou k němu připojené. Když se podíváme do směrovací tabulky směrovače R6, uvidíme, že má statickou cestu k sítím 192.168.0.0/21 a další hop pro něj je rozhraní eth1 na směrovači R9. Abychom mohli tuto statickou cestu redistribuovat, je potřebné nejdříve vytvořit prefix-list a route-mapu na směrovači R6. Prefix-list a route-mapu vytvoříme pomocí následujících příkazů v konfiguračním módu:

set policy prefix-list R9-net-preflist rule 1 prefix 192.168.0.0/21
set policy prefix-list R9-net-preflist rule 1 action permit
set policy route-map R9-net-rmap rule 10 match ip address prefix-list R9-net-preflist
set policy route-map R9-net-rmap rule 10 action permit
commit

Potom pomocí vytvořené route-mapy redistribuujeme statickou síť do OSPF pomocí příkazu:

set protocols ospf redistribute static route-map R9-net-rmap
commit
save
exit

Pro kontrolu si můžeme zobrazit směrovací tabulku libovolného směrovače. Jako příklad je uvedena směrovací tabulka směrovače R1 na obrázku 8-22.

```
student@R1:~$ show ip route
Codes: K - kernel route, C - connected, S - static, R - RIP,
       O - OSPF, I - IS-IS, B - BGP, E - EIGRP, N - NHRP,
       T - Table, v - VNC, V - VNC-Direct, A - Babel, D - SHARP,
       F - PBR, f - OpenFabric,
       > - selected route, * - FIB route, q - queued route, r - rejected route

S>* 0.0.0.0/0 [210/0] via 192.168.122.1, eth0, 00:17:02
O   10.0.0.0/29 [110/100] is directly connected, eth1, 00:16:15
C>* 10.0.0.0/29 is directly connected, eth1, 00:17:09
O>* 10.10.0.0/16 [110/300] via 10.0.0.3, eth1, 00:16:02
O>* 10.20.0.0/16 [110/300] via 10.0.0.3, eth1, 00:16:03
O>* 10.30.0.0/16 [110/300] via 10.0.0.2, eth1, 00:15:55
O>* 192.168.0.0/21 [110/20] via 10.0.0.2, eth1, 00:00:24
O>* 192.168.32.0/22 [110/210] via 10.0.0.3, eth1, 00:16:03
O>* 192.168.36.0/22 [110/210] via 10.0.0.3, eth1, 00:16:02
C>* 192.168.122.0/24 is directly connected, eth0, 00:17:03
```

Obr. 8-22: Směrovací tabulka směrovače R1

Zde můžeme vidět, že do směrovací tabulky přibyla IP adresa 192.168.0.0/21. Na směrovači R9 se nachází loopback rozhraní s IP adresami 192.168.0-7.1 a taky se tam nachází virtuální počítač s IP adresou 192.168.1.10. Konzoli počítače si můžeme spustit dvojklikem na počítač a vyzkoušet, jestli počítač skutečně dokáže komunikovat se všemi zařízeními v síti (kromě zařízení nacházejících se v oblasti 51).

8.3.3 Virtuální link

Virtuální link se v OSPF sítích využívá v případě, že nějaká oblast není přímo spojená s oblastí 0. Jako příklad se dá použít topologie z této laboratorní úlohy, ve které oblast 51 není přímo spojená s oblastí 0, a tak nemůže komunikovat v síti. V takovém případě se vytváří virtuální link mezi hraničními směrovači, v našem případě to je mezi směrovači R2 a R7. Že směrovače v této oblasti skutečně nemají informace o sítích v ostatních oblastech, si můžeme ověřit tak, že se podíváme do směrovací tabulky směrovače R8. Směrovací tabulka směrovače je na obrázku 8-23.

```
student@R8:~$ show ip route
Codes: K - kernel route, C - connected, S - static, R - RIP,
       O - OSPF, I - IS-IS, B - BGP, E - EIGRP, N - NHRP,
       T - Table, v - VNC, V - VNC-Direct, A - Babel, D - SHARP,
       F - PBR, f - OpenFabric,
       > - selected route, * - FIB route, q - queued route, r - rejected route

O   10.51.0.0/30 [110/100] is directly connected, eth0, 00:01:29
C>* 10.51.0.0/30 is directly connected, eth0, 00:01:38
O   10.51.1.0/24 [110/100] is directly connected, eth1, 00:01:29
C>* 10.51.1.0/24 is directly connected, eth1, 00:01:39
O   10.51.2.0/24 [110/10] is directly connected, dum2, 00:01:29
C>* 10.51.2.0/24 is directly connected, dum2, 00:01:42
O   10.51.3.0/24 [110/10] is directly connected, dum3, 00:01:29
C>* 10.51.3.0/24 is directly connected, dum3, 00:01:41
```

Obr. 8-23: Směrovací tabulka směrovače R8

Na obrázku můžeme vidět, že směrovač obsahuje pouze sítě, které se nachází přímo v oblasti 51. Taky jsme mohli vidět v přecházejících tabulkách, že ostatní směrovače nemají směrovací informace o sítích v oblasti 51. Abychom oblast 51 spojili s ostatními oblastmi, vytvoříme virtuální link v průchozí oblasti (oblast 30). Do směrovače mezi oblastmi 0 a 30 (to jest směrovač R2) zadáme následující příkazy:

configure
set protocols ospf area 30 virtual-link 7.7.7.7
commit
save
exit

Do příkazu pro virtuální link se zadává průchozí oblast (v našem případě 30) a Router-ID cílového směrovače. V našem případě je Router-ID směrovače R7 nastavené na 7.7.7.7. Obdobné nastavení jako u směrovače R2 je potřebné nastavit i na směrovači R7 a hned nakonfigurujeme i sumarizaci sítě 10.51.0.0/16:

configure
set protocols ospf area 30 virtual-link 2.2.2.2
set protocols ospf area 51 range 10.51.0.0/16
commit
save
exit

Po zadání příkazů si můžeme na směrovači R7 pomocí příkazu *show ip ospf neighbor* zkontrolovat, jestli se nám vytvořilo sousedství mezi směrovačem R2 a R7. Výstup by měl vypadat jako na obrázku 8-24.

```
student@R7:~$ show ip ospf neighbor

Neighbor ID     Pri State           Dead Time Address         Interface                        RXmtL RqstL DBsmL
6.6.6.6           1 Full/Backup       35.244s 10.30.0.5       eth2:10.30.0.6                       0     0     0
8.8.8.8           1 Full/DR           30.514s 10.51.0.2       eth0:10.51.0.1                       0     0     0
2.2.2.2           1 Full/DROther      35.234s 10.30.0.1       VLINK0                               0     0     0
```

Obr. 8-24: OSPF sousedi směrovače R7

Ve výstupu můžeme vidět, že je vytvořeno sousedství se směrovačem R2 přes rozhraní VLINK0. Teď si můžeme ověřit, že na směrovači R8 už máme úplně celou tabulku OSPF sítí. Směrovací tabulka směrovače R8 by měla vypadat jako na obrázku 8-25.

```
student@R8:~$ show ip route
Codes: K - kernel route, C - connected, S - static, R - RIP,
       O - OSPF, I - IS-IS, B - BGP, E - EIGRP, N - NHRP,
       T - Table, v - VNC, V - VNC-Direct, A - Babel, D - SHARP,
       F - PBR, f - OpenFabric,
       > - selected route, * - FIB route, q - queued route, r - rejected route

O>* 0.0.0.0/0 [110/101] via 10.51.0.1, eth0, 00:01:42
O>* 10.0.0.0/29 [110/400] via 10.51.0.1, eth0, 00:01:23
O>* 10.10.0.0/16 [110/600] via 10.51.0.1, eth0, 00:01:23
O>* 10.20.0.0/16 [110/600] via 10.51.0.1, eth0, 00:01:23
O>* 10.30.0.0/16 [110/500] via 10.51.0.1, eth0, 00:01:23
O>* 10.30.0.0/30 [110/300] via 10.51.0.1, eth0, 00:01:42
O>* 10.30.0.4/30 [110/200] via 10.51.0.1, eth0, 00:01:42
O>* 10.30.1.0/30 [110/300] via 10.51.0.1, eth0, 00:01:42
O   10.51.0.0/30 [110/100] is directly connected, eth0, 00:09:07
C>* 10.51.0.0/30 is directly connected, eth0, 00:09:16
O   10.51.1.0/24 [110/100] is directly connected, eth1, 00:09:07
C>* 10.51.1.0/24 is directly connected, eth1, 00:09:17
O   10.51.2.0/24 [110/10] is directly connected, dum2, 00:09:07
C>* 10.51.2.0/24 is directly connected, dum2, 00:09:20
O   10.51.3.0/24 [110/10] is directly connected, dum3, 00:09:07
C>* 10.51.3.0/24 is directly connected, dum3, 00:09:19
O>* 192.168.32.0/22 [110/510] via 10.51.0.1, eth0, 00:01:23
O>* 192.168.36.0/22 [110/510] via 10.51.0.1, eth0, 00:01:23
```

Obr. 8-25: Směrovací tabulka směrovače R8 po přidání virtuálního linku

Nyní máme na směrovači R8 v jeho směrovací tabulce úplně všechny sítě, které se nacházejí v celé OSPF síti. Ale jelikož směrovač R8 má jen jednu výchozí bránu ze sítě, a to konkrétně směrovač R7, tak mu stačí, že má ve směrovací tabulce jen sítě nacházející se v oblasti 51, o ostatních sítích nepotřebuje mít žádné informace. To se dá vyřešit pomocí takzvaných stub oblastí.

8.3.4 Stub oblast

V této podkapitole chceme dosáhnout toho, aby směrovač R8 měl informace jen o své vlastní oblasti, kterou je oblast 51. Z této oblasti je jen jedna výchozí brána, a to směrovač R7, tudíž ostatní oblasti směrovač R8 nezajímají. Abychom dosáhli toho, že oblast bude typu stub, musíme tento typ nastavit na všech směrovačích v oblasti. V našem případě máme směrovače jen dva, a to R7 a R8. Na směrovači R7 musíme oblast nastavit jako totally stub, aby do dané oblasti neposílal žádné vnější směrovací informace.

Do směrovače R7 vložíme následující příkazy:

configure
set protocols ospf area 51 area-type stub no-summary
commit
save
exit

Stačí, když do směrovače R8 nastavíme, že je to stub oblast. Nemusí být nastavený jako no-summary:

configure
set protocols ospf area 51 area-type stub
commit
save
exit

Kdybychom teď zkontrolovali směrovací tabulku směrovače R8, byly by tam pořád všechny adresy, které se nachází v OSPF síti. Proto, abychom viděli nějaký výsledek, musíme restartovat OSPF proces. Pro restart procesu OSPF použijeme následující příkaz na směrovači R8:

sudo ${vyos_op_scripts_dir}/restart_frr.py --action restart --daemon ospfd

Směrovač nás upozorní na to, že je to potenciálně nebezpečný příkaz, proto je potřebné potvrdit jej písmenem „y“ a stisknout enter. Poté bude chvíli trvat, než se do směrovače R8 načte směrovací tabulka. Po dokončení by měla směrovací tabulka vypadat jako na obrázku 8-26.

```
student@R8:~$ show ip route
Codes: K - kernel route, C - connected, S - static, R - RIP,
       O - OSPF, I - IS-IS, B - BGP, E - EIGRP, N - NHRP,
       T - Table, v - VNC, V - VNC-Direct, A - Babel, D - SHARP,
       F - PBR, f - OpenFabric,
       > - selected route, * - FIB route, q - queued route, r - rejected route

O>* 0.0.0.0/0 [110/101] via 10.51.0.1, eth0, 00:00:54
O   10.51.0.0/30 [110/100] is directly connected, eth0, 00:01:49
C>* 10.51.0.0/30 is directly connected, eth0, 00:19:31
O   10.51.1.0/24 [110/100] is directly connected, eth1, 00:01:49
C>* 10.51.1.0/24 is directly connected, eth1, 00:19:32
O   10.51.2.0/24 [110/10] is directly connected, dum2, 00:01:49
C>* 10.51.2.0/24 is directly connected, dum2, 00:19:33
O   10.51.3.0/24 [110/10] is directly connected, dum3, 00:01:49
C>* 10.51.3.0/24 is directly connected, dum3, 00:19:32
```

Obr. 8-26: Směrovací tabulka směrovače R8

Směrovač obsahuje pouze směrovací informace oblasti, ve které se nachází, a výchozí bránu.

8.3.5 Pasivní rozhraní

U protokolu OSPF je možné nastavit takzvané pasivní rozhraní. To slouží k tomu, aby směrovač do daného rozhraní neposílal Hello pakety, ale i přesto síť propagoval v OSPF síti. Nastavuje se například na rozhraních, které vedou ke koncovým zařízením, protože ta nepotřebují mít informaci o tom, že se v síti využívá OSPF protokol, ani nepotřebují navazovat OSPF sousedství se směrovačem.

Způsoby, jak nakonfigurovat pasivní rozhraní, jsou dva. První je, že každé rozhraní jednotlivě nastavíme jako pasivní. Druhý způsob je, že jedním příkazem se nastaví všechna rozhraní jako pasivní a potom budeme rozhraní, která nemají být pasivní, odebírat.

V naší topologii využijeme oba způsoby konfigurace. Směrovače R4 a R5 nakonfigurujeme druhým způsobem a směrovače R6 a R8 prvním způsobem.

Pro nakonfigurování všech rozhraní jako pasivních a potom pro vyloučení konkrétních rozhraní použijeme tyto příkazy na směrovači R4:

configure
set protocols ospf passive-interface default
set protocols ospf passive-interface-exclude eth0
commit
save
exit

Pro směrovač R5 budou příkazy stejné, rozdíl bude jen v tom, že tam chceme z pasivních rozhraní vyloučit rozhraní eth1.

Pro nastavení každého rozhraní jako pasivního zadáme jednotlivě tyto příkazy do směrovače R6:

configure
set protocols ospf passive-interface eth1
commit
save
exit

Pro směrovač R8 budou příkazy stejné. Zkontrolujte si jen, na kterém rozhraní chcete nastavovat pasivní rozhraní. Když bude pasivní rozhraní nastaveno směrem k dalšímu směrovači s OSPF, tak se mezi těmito směrovači rozváže jejich sousedství. Na směrovači si můžeme zobrazit informace o rozhraní, že je skutečně nastaveno jako pasivní pomocí příkazu *show ip ospf interface <název rozhraní>*. Příklad výstupu ze směrovače R8 na rozhraní eth1 vidíme na obrázku 8-27.

```
student@R8:~$ show ip ospf interface eth1
eth1 is up
  ifindex 3, MTU 1500 bytes, BW 1000 Mbit <UP,BROADCAST,RUNNING,MULTICAST>
  Internet Address 10.51.1.1/24, Broadcast 10.51.1.255, Area 0.0.0.51 [Stub]
  MTU mismatch detection: enabled
  Router ID 8.8.8.8, Network Type BROADCAST, Cost: 100
  Transmit Delay is 1 sec, State DR, Priority 1
  No backup designated router on this network
  Multicast group memberships: <None>
  Timer intervals configured, Hello 10s, Dead 40s, Wait 40s, Retransmit 5
    No Hellos (Passive interface)
  Neighbor Count is 0, Adjacent neighbor count is 0
```

Obr. 8-27: Informace o OSPF rozhraní

V předposledním řádku vidíme, že rozhraní je pasivní a neposílají se do něj Hello pakety.

8.4 Reakce na změnu v síti

Když v OSPF nastane změna v síti (nastane výpadek, přibude nová síť a podobně), tak se po celé OSPF síti posílají LSA (Link State Advertisement) pakety, které informují o změnách v síti, a směrovače si podle nich upravují svou směrovací tabulku.

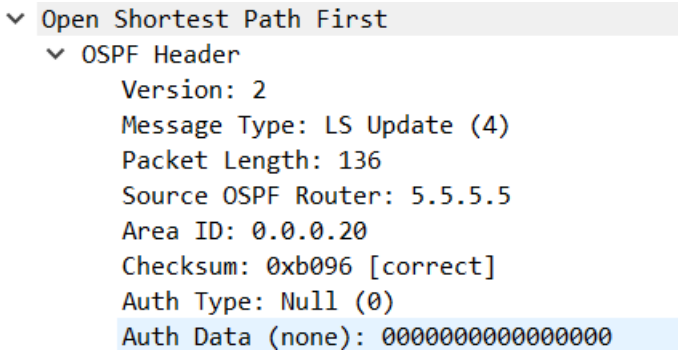
V této části laboratorní úlohy si zkusíme simulovat výpadek v síti a odchytíme si LSA pakety.

Pro příklad si odsimulujeme výpadek v oblasti 20 tím, že počítač PC2 odpojíme od sítě. To uděláme tak, že vypneme rozhraní eth2 na směrovači R5. Předtím si zapneme zachytávání paketů mezi směrovačem R5 a R3. Rozhraní eth2 vypneme pomocí příkazu v konfiguračním módu:

set interfaces ethernet eth2 disable

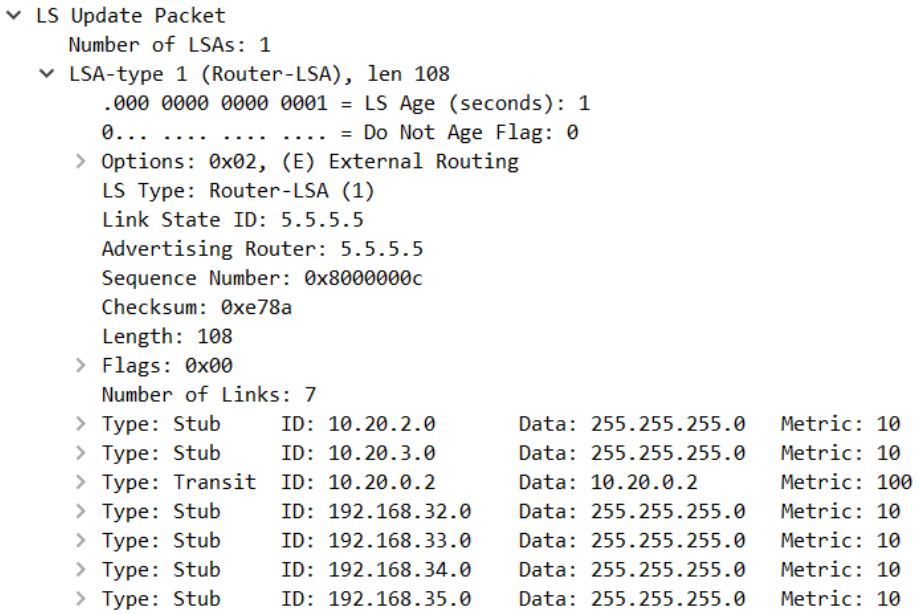
Potvrdíme příkazem *commit* a uložíme příkazem *save*.

Když se podíváme do Wiresharku na odchycené pakety, můžeme vidět, že se přenesly dva pakety, LS Update a LS Acknowledge. Když si rozklikneme paket LS Update, vidíme, že se skládá ze dvou částí, OSPF Header a LS Update Packet. Na obrázku 8-28 můžeme vidět OSPF Header.



Obr. 8-28: OSPF header

V OSPF header se nachází verze OSPF, typ OSPF zprávy, velikost paketu, OSPF Router-ID, OSPF oblast, kontrolní součet a autentifikace. Teď se podíváme na druhou část OSPF zprávy. Ta je zobrazená na obrázku 8-29. Na obrázku můžeme vidět, že mezi sítěmi se nenachází síť 10.10.1.0, což je síť, která je připojená na rozhraní eth1 na směrovači R5. To je způsobeno tím, že se směrovač R3 naučil, že tato síť není dostupná, a vymazal si ji ze směrovací tabulky.

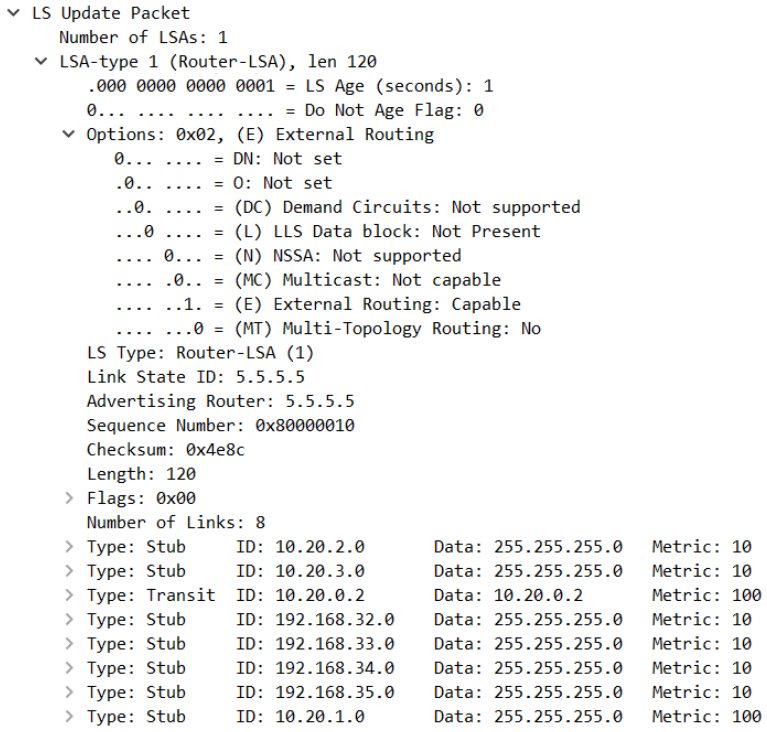


Obr. 8-29: LS Update Packet

Nyní rozhraní na směrovači R5 znovu zapneme. To uděláme následujícím příkazem v konfiguračním módu a potvrdíme konfiguraci:

delete interfaces ethernet eth2 disable

Ve Wireshark se nám odchytila další dvojice LS Update a LS Acknowledge. Když si rozklikneme LS Update a podíváme se do druhé části paketu, která je zobrazená na obrázku 8-30, můžeme vidět, že teď se tam nachází síť 10.20.1.0. Tím si směrovač R3 může aktualizovat svou směrovací tabulku.



Obr. 8-30: LS Update Packet po zapnutí rozhraní

8.5 Samostatní úkol

V předchozí úloze jsme si popsali některé funkce, které jsou obsaženy v protokolu OSPF. Zkuste ve směrovačích zjistit, které funkce se nacházejí také v protokolu RIP a které se v něm nenacházejí.

To zjistíme tak, že vstoupíme do konfiguračního módu směrovače pomocí příkazu *configure,* zadáme příkaz *set protocols rip* a stlačíme klávesu tab nebo otazník, směrovač nám tam doporučí možné příkazy.

8.6 Kontrolní otázky

- Na základě čeho protokol RIP/OSPF počítá metriku?
- Je možné u protokolu RIP změnit výběr nejkratší cesty, jako to jde v případě OSPF pomocí šířky pásma nastavené na rozhraní?
- Protokol RIP má nastavený maximální počet skoků na 15. Je možné toto číslo změnit (zkuste zjistit v konfiguračním módu směrovače)?
- Kdy protokol RIP/OSPF dělá aktualizace?
- K čemu slouží Hello paket?
- Jsou u protokolu OSPF při aktualizaci posílány kompletní směrovací tabulky?
- Dokáže RIPv2/OSPF pracovat s classless adresami?
- Které směrovací protokoly ještě znáte (kromě RIP a OSPF)?

9. ZÁVER

Táto diplomová práca sa zaoberá vytvorením dvoch laboratórnych úloh pre študentov bakalárskeho štúdia. Prvá laboratórna úloha sa zaoberá porovnaním transportných protokolov TCP a UDP a druhá laboratórna úloha sa zaoberá porovnaním smerovacích protokolov OSPF a RIP.

V teoretickej časti práce je popísaný postup na inštaláciu použitých virtuálnych strojov a tiež je tam opísaný použitý operačný systém VyOS používaný v smerovačoch. Ďalej sú v teoretickej časti popísané samotné transportné a smerovacie protokoly.

Prvá laboratórna úloha zameraná na porovnanie transportných protokolov je zložená zo štyroch scenárov. Prvý scenár je základné porovnanie protokolov. Študenti sa v prvom scenári zoznámia s pracovným prostredím GNS3 a tiež si vyskúšajú základnú konfiguráciu Linux servera s operačným systémom CentOS. Ďalej si v prvom scenári odchytia jednoduchú TCP a UDP komunikáciu a môžu pozorovať, že pri TCP protokole sa najprv nadväzuje spojenie až potom sa posielajú dáta, naopak pri protokole UDP sa dáta posielajú bez nadväzovania spojenia. V ďalšom scenári je simulovanie výpadku linky a je sledované ako sa protokoly správajú, keď linka vypadne. Pri protokole TCP je vidno, že sa dáta prestali posielať a protokol sa snažil spojenie obnoviť, čo sa mu aj podarilo po tom ako sa linka znova zapla. Keďže protokol UDP nemá žiadny mechanizmus na to aby zaznamenal, že linka vypadla, tak sa dáta posielali ďalej. Ďalší scenár je simulovanie zahadzovania paketov. V tomto scenári môžu študenti sledovať, že ak bolo nastavené väčšie zahadzovanie paketov, tak sa zahodené dáta museli znova poslať a tým pádom bolo viac prenesených dát. V poslednom scenári je simulované zahltenie linky. V tomto scenári môžu študenti sledovať, že čím viac linku zahltia tým na dlhšiu dobu obmedzia prenos TCP paketov.

Druhá laboratórna úloha je zameraná na smerovacie protokoly. V prvej časti je základné porovnanie protokolov RIP a OSPF. Ďalej je v úlohe viac dopodrobna rozobraný protokol OSPF. Sú tam vysvetlené jeho funkcie ako sú sumarizácia sieti, redistribúcia z iných smerovacích protokolov do OSPF, virtuálny link, stub oblasť, pasívne rozhrania a reakcia OSPF na zmenu v sieti. Študenti sa v tejto úlohe zoznámia so so simulačným prostredím GNS3, so základnou konfiguráciou smerovačov s operačným systémom VyOS a so smerovacími protokolmi RIP a OSPF.

Na konci každej laboratórnej úlohy sa nachádzajú jednoduché kontrolné otázky z laboratórnych úloh.

Literatura

[1] GNS3 Documantation [online]. [cit. 2019-10-27]. Dostupné z: https://docs.gns3.com/

[2] Introduction — ns-3 project ns-3-dev documentation. Ns-3 | a discrete-event network simulator for internet systems [online]. [cit. 2019-12-15]. Dostupné z: https://www.nsnam.org/docs/tutorial/html/introduction.html

[3] Cisco IOS images for Dynamips. *GNS3* [online]. [cit. 2019-12-02]. Dostupné z: https://docs.gns3.com/1-kBrTplBltp9P3P-AigoMzlDO-ISyL1h3bYpOl5Q8mQ/index.html

[4] *Wireshark – About* [online]. [cit. 2019-10-28]. Dostupné z: https://www.wireshark.org/about.html

[5] *Wireshark – Introduction* [online]. [cit. 2019-10-28]. Dostupné z: https://www.wireshark.org/docs/wsug_html_chunked/ChapterIntroduction.html

[6] VyOS User Guide [online]. [cit. 2020-05-01]. Dostupné z: https://docs.vyos.io

[7] JEŘÁBEK, J. Komunikační technologie. Skriptum FEKT Vysoké učení technické v Brně, 2016. s. 1-172.

[8] POSTEL, J. RFC 768: User Datagram Protocol [online]. 1980 [cit. 2019-12-14]. Dostupné z: https://tools.ietf.org/html/rfc768

[9] FOROUZAN, Behrouz A. TCP/IP protocol suite. 4th ed. Boston: McGraw-Hill Higher Education, 2010. ISBN 978-0-07-337604-2.

[10] RFC 793: TRANSMISSION CONTROL PROTOCOL [online]. 1981 [cit. 2019-03-13]. Dostupné z: https://tools.ietf.org/html/rfc793#section-3.1

[11] GRYGÁREK, Petr. Směrovací protokol OSPF [online]. [2004] , změněno 1. listopadu 2004 [cit. 2020-04-21]. Kódováno v ISO-8859-2. Text v češtině. Dostupný z WWW: http://www.cs.vsb.cz/grygarek/SPS/lect/OSPF/ospf.html

[12] IP směrování. Mendelova Univerzita v Brně [online]. [cit. 2020-05-01]. Dostupné z: https://is.mendelu.cz/eknihovna/opory/zobraz_cast.pl?cast=599

Príloha 1 - príloha v zip súbore

- Priečinok TCP-UDP – Obsahuje konfigurácie smerovačov R1 a R2 a skripty použité vo virtuálnych strojoch Client a Server
- Priečinok OSPF – obsahuje základnú konfiguráciu smerovačov R1 až R9 a počítačov PC1 až PC4